VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Rozpoznanie objektov a ich polohy s využitím 3D modelov objektov

# Object and Pose Recognition Using 3D Object Models

2017 Patrik Herák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání diplomové práce

Student: **Bc. Patrik Herák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozpoznání objektů a jejich polohy s využitím 3D modelů objektů
Object and Pose Recognition Using 3D Object Models

Jazyk vypracování: čeština

Zásady pro vypracování:

Rozpoznávání objektů na základě 3D modelů může hrát významnou roli v úlohách rozšířené reality, kdy často, s ohledem na předpokládaný větší počet typů objektů, nelze využít postupů založených na klasickém učení využívajícím většího množství možných pohledů na 3D objekt. V diplomové práci proveďte následující:

1. Seznamte se přiměřeně s kontextem vaší budoucí práce (rozpoznávání na základě 3D modelu). Seznamte se s knihovnou [1].
2. S využitím uvedené knihovny realizujte software pro rozpoznávání objektů z 3D map. Vyhodnoťte získané zkušenosti.
4. Na základě zkušeností dle předchozího bodu uvažte možnost modifikace vybraných metod z knihovny s cílem dosáhnout urychlení rozpoznání (nejlépe v reálném čase) nebo s cílem zvětšení přesnosti rozpoznání.
5. Realizovaný software důkladně experimentálně ověřte. Popište nedostatky (a také přednosti) zvoleného postupu. Zvažte, zda a jak by případné nedostaky bylo možné odstranit.
4. Vše pečlivě zdokumentujte v textové části práce. Implementaci proveďte v C/C++.

Seznam doporučené odborné literatury:

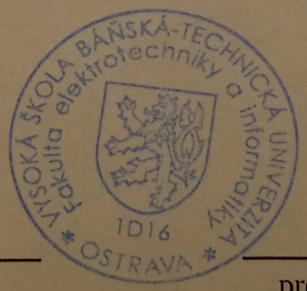
[1] Point Cloud Library: http://pointclouds.org/

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Dr. Ing. Eduard Sojka**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017




doc. Dr. Ing. Eduard Sojka
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa 24.4.2017

.....

Rád by som na tomto mieste poďakoval doc. Dr. Ing. Eduardovi Sojkovi za vedenie diplomovej práce. V neposlednom rade by som chcel poďakovať za podporu mojej priateľke a blízkym, pretože bez nich by táto práca nevznikla.

**Abstrakt**

Práca sa zaoberá detekciou objektov a ich 3D pozície v RGB-D obraze (získaného zo senzoru Microsoft Kinect) na základe 3D modelov. Cieľom práce bolo vytvorenie aplikácie slúžiacej k rozpoznaniu objektu v obraze s použitím rôznych State of the Art algoritmov určených k popisu okolia bodu (tzv. lokálne deskriptory), na základe ktorých sa hľadá objekt v obraze. Následne boli tieto algoritmy vyhodnotené a boli navrhnuté a experimentálne overené možné vylepšenia tohto procesu. Navrhnuté riešenia zahrnujú redukciu dimenzií odhadnutých deskriptorov pomocou analýzy hlavných komponentov, pridanie farebnej zložky do lokálneho deskriptoru Fast Point Feature Histogram (FPFH), modifikovanie deskriptoru FPFH tak, aby bol reprezentovaný menším počtom čísel. Poslednou fázou je porovnanie efektívnosti algoritmov použitých pre vyhľadávanie najbližších susedných bodov.

**Kľúčové slová**: rozpoznávanie objektov, mračno bodov, Microsoft Kinect, normály, lokálne deskriptory, Fast Point Feature Histogram (FPFH), analýza hlavných komponentov (PCA), k-d strom

**Abstract**

Thesis addresses one of the typical tasks in computer vision, called 3D pose estimation. Object recognition is the fundamental task for different areas, such as robotics or augumented reality. The goal of this thesis was to develop a software for 3D pose estimation in RGB-D image based on object's model using various State of the Art local descriptors algorithms. Later, these algorithms are compared and evaluated. Moreover, a few enhancements are proposed to make object recognition process more effective. These include dimension reduction using principal component analysis (PCA), adding color to the Fast Point Feature Histogram (FPFH) descriptor, modifying FPFH descriptor to estimate low dimensional descriptors and comparing algorithms for nearest neighbor search.

**Key Words**: object detection, point cloud, Microsoft Kinect, normals, local descriptors, Fast Point Feature Histogram (FPFH), principal component analysis (PCA), k-d tree

# Obsah

# Zoznam použitých skratiek a symbolov

| | | |
|---|---|---|
| FLANN | – | Fast Library for Approximate Nearest Neighbors |
| FPFH | – | Fast Point Feature Histogram |
| PCA | – | Principal Component Analysis |
| PCL | – | Point Cloud Library |
| PFH | – | Point Feature Histogram |
| PFHRGB | – | Point Feature Histogram s farebnou zložkou |
| SHOT | – | Signature of Histograms of OrienTations |
| SHOTCOLOR | – | Signature of Histograms of OrienTations s farebnou zložkou |
| VFH | – | Viewpoint Feature Histogram |
| VTK | – | Visualization Toolkit |

# Zoznam obrázkov

# Zoznam tabuliek

# Zoznam výpisov zdrojového kódu

# 1 Úvod

**Počítačové videnie** je veda, ktorá sa zaoberá tým, ako naučiť počítače rozumieť obrazu (digitálnym obrázkom alebo videám). Vznikla v šesťdesiatych rokoch minulého storočia na univerzitách, ktoré skúmali ďalší príbuzný obor - **umelú inteligenciu**.

Od tej doby sa disciplína posunula o veľký krok vpred a použitia môžeme vidieť v našich každodenných životoch. V priemyselnej sfére sa veľmi často stretávame s aplikáciou počítačového videnia na detekciu chybných kusov vo výrobnej linke alebo na meranie pozície a orientácie vecí, s ktorými bude manipulovať robotická ruka. Takéto použitia sa často označujú ako **strojové videnie**. Ďalším využitím je napríklad detekcia udalostí (počítanie ľudí, voľných parkovacích miest), alebo autonómne šoférovanie.

Počítačové videnie blízko súvisí s ďalšou, čím ďalej rozšírenejšou témou, **rozšírená realita**. Dôvodom je fakt, že cena snímacích zariadení, ako napríklad Microsoft Kinect alebo Asus Xtion PRO, rapídne klesla. Vďaka tomu sa snímanie 3D obrazu stalo dostupnejším, čím sa rozšírila aj komunita vedcov a vývojárov, ktorí sa zaujímajú o spracovanie 3D obrazu. Rozšírená realita nám umožňuje vložiť do obrazu, ktorý je zachytený kamerou, virtuálne objekty. Systém rozšírenej reality by mal splňovať tieto vlastnosti [6]:

- kombinovať reálne a virtuálne objekty
- byť interaktívny v reálnom čase
- byť schopný registrovať reálne a virtuálne objekty

Základnou úlohou pri týchto aplikáciách je hľadanie pozície objektu v zachytenej scéne. Vďaka získanej pozícii môžeme napríklad pridávať do obrazu spomínané virtuálne objekty.

Preto cieľom tejto práce je v prvom kroku vytvorenie aplikácie na rozpoznávanie objektov v RGB-D obraze získaného zo senzoru Microsoft Kinect za použitia rôznych State of the Art algoritmov pre popis geometrie okolia bodu (tzv. 3D deskriptory). K tomu som použil knižnicu Point Cloud Library (PCL), ktorá slúži pre spracovanie 2D/3D obrazu a mračien bodov. Celý proces vývoja aj s vyhodnotením úspešnosti a použiteľnosti testovaných algoritmov v praxi je popísaný v kapitole 4.

Druhá časť práce (viď kapitola 5) je venovaná návrhom a ich experimentálnym overeniam. Tieto návrhy boli vymyslené k vylepšeniu procesu rozpoznávania objektov za účelom zrýchlenia a spresnenia výsledkov. V praxi sú deskriptory reprezentované veľkým počtom čísel, čo má za následok zníženie efektívnosti niektorých algoritmov. Preto mojim prvým vylepšením je redukcia dimenzií vybraných deskriptorov pomocou analýzy hlavných komponentov za účelom zrýchlenia procesu rozpoznávania objektu s následným vyhodnotením presnosti po redukcii (viď kapitola 5.1).

Druhou zmenou je modifikácia deskriptoru Fast Point Feature Histogram (FPFH), ktorý je implementovaný v knižnici PCL. Modifikácia spočíva v tom, aby sa už pri samotnom odhade

používal menší počet čísel pre jednotlivé histogramy, ktoré tvoria deskriptor (viď kapitola 5.2), čo má za následok redukciu dimenzií priamo pri výpočte.

Následne je implementácia algoritmu FPFH rozšírená o farebnú zložku RGB. Takto modifkovaný deskriptor som nazval FPFHCOLOR. Detailnému popisu a dosiahnutým výsledkom je venovaná kapitola 5.3. Na tento deskriptor je v ďalšom kroku aplikovaná analýza hlavných komponentov za účelom zistenia toho či je farebná zložka naozaj potrebná.

Ako posledné som pre potreby porovnania efektívnosti vyhľadávacích algoritmov založených na hľadaní najbližších susedov (napr. k-d strom) naprogramoval algoritmus, ktorý hľadá najbližších susedov tzv. hrubou silou. Motiváciou pre tvorbu tohto algoritmu bola prípadná budúca paralelizácia na GPU. Kapitola 5.4 obsahuje porovnanie času s bežne používaným k-d stromom pri sériovom spracovaní.

# 2 State of the Art

Rozpoznávanie objektov v scénach je základná výskumná oblasť v počítačovom videní. Má veľa využití, ako napríklad inteligentné pozorovacie systémy, automatické skladanie v priemysle, robotika, biometrická analýza a zdravotné ošetrenie.

V posledných desaťročiach bola dôkladne preskúmaná výskumná oblasť zaoberajúca sa rozpoznávaním objektov v 2D obraze. V porovnaní s 2D obrazom má 3D obraz niekoľko výhod pre rozpoznanie objektov, napríklad:

1. poskytuje viac geometrických informácií vďaka hĺbke,
2. deskriptory získané z 3D obrazu nie sú väčšinou ovplyvňované mierkou, rotáciou a osvetlením,
3. rozpoznaná 3D pozícia objektu je presnejšia ako z 2D obrazu.

3D obraz má preto potenciál prekonať mnoho nedostatkov 2D obrazu v súvislosti s rozpoznávaním objektov. Tieto výhody prispeli k tomu, že sa z rozpoznávania objektov v 3D obraze stala aktívna výskumná oblasť. Navyše rýchly vývoj nízkonákladových snímacích zariadení, ako napríklad Microsoft Kinect alebo Asus Xtion PRO, spravilo snímanie 3D obrazu dostupnejším.

Okrem toho, pokroky v počítačovej technike dovoľujú spracovávanie náročnejších algoritmov pre rozpoznávanie objektov v 3D obraze v rozumnom čase.

Všetky tieto faktory prispeli k rozšíreniu výskumu zameriavajúcemu sa na vývoj systémov na rozpoznávanie objektov v 3D obraze.

Existujúce metódy rozpoznávania objektov v 3D obraze môžeme rozdeliť do dvoch kategórií:

1. metódy založené na **lokálnych deskriptoroch**,
2. metódy založené na **globálnych deskriptoroch**.

Celý postup hľadania objektu v 3D scéne na základe modelu vyhľadávaného objektu sa skladá z niekoľkých krokov, ktoré sú detailnejšie popísané v kapitole 3.

Pre popis okolia bodov sa používajú algoritmy, ktoré nazývame **deskriptory** (viď kapitola 3.4). Lokálne deskriptory dosahujú lepšie výsledky v preplnených scénach a nie sú závislé na segmentácii obrazu, ako globálne deskriptory. Okrem toho je známe, že pomocou globálnych deskriptorov nie je možné rozpoznávať čiastočne viditeľné objekty. Medzi najpoužívanejšie deskriptory v tejto kategórii aktuálne patria:

- Signatures of Histograms of OrienTations (SHOT)
- Point Feature Histogram (PFH)
- Fast Point Feature Histogram (FPFH)

Aj z týchto dôvodov je moja práca zameraná na použitie a vylepšenie lokálnych deskriptorov.

# 3 Teoretická časť

Pozícia pevného telesa má v priestore šesť stupňov voľnosti (6 DoF alebo 6D) [11], pretože je možné určiť tri nezávislé súradnice (x,y,z) jeho ťažiska a môže sa natočiť okolo troch nezávislých osí. Poznanie takto definovanej pozície objektu je väčšinou v praxi nanajvýš potrebné.

V robotike 6D pozícia objektu napomáha priestorovému uvažovaniu robota a dovoľuje mu manipulovať s ním. Pri použití v aplikáciach rozšírenej reality je možné vďaka pozícii rozširovať objekty o ďalšie informácie, ako napríklad rady pri skladaní / rozkladaní rôznych častí áut, lietadiel a podobných väčších celkov.

Ideálne by mal byť systém na rozpoznávanie 3D objektov schopný získať mračná bodov zo snímacích zariadení (získanie scény), vypočítať deskriptory, porovnať ich s tými, ktoré sú uložené v databáze objektov (modely rozpoznávaných objektov) a nájsť všetky zhody s ich pozíciami a orientáciami v scéne, v reálnom čase.

K dosiahnutiu požadovaného výsledku (získanie čo najpresnejšej 6D pozície objektu) je treba vykonať niekoľko krokov, ktoré budú postupne popísané v tejto kapitole. Teoretický popis jednotlivých krokov a algoritmov v tejto kapitole je dôležitý, pretože budú prakticky využité pri vývoji softwaru, ktorý je určený na rozpoznávanie objektu (na základe jeho modelu) v RGB-D obraze a ktorý je súčasťou tejto práce (viď kapitola 4). Následne sú niektoré teoretické podklady využité pri experimentoch, ktorým je venovaná kapitola 5. Väčšina obrázkov v tejto kapitole je prevzatých z [13].

Ako prvá je predstavená základná dátová štruktúra používaná pri práci s 3D obrazom, ktorá sa nazýva **mračno bodov** (viď kapitola 3.1). Nasleduje detailný popis výpočtu **povrchových normál**, ktoré reprezentujú základ pre popis geometrie v okolí bodu (viď kapitola 3.2). Po tomto nevyhnutnom základe nasleduje detailný popis tzv. **deskriptorov**, ktoré umožňujú detailnejší popis geometrie v okolí bodu (viď kapitola 3.4). V praxi je výpočet deskriptorov pre každý bod v mračne výpočtovo náročný, preto sa z mračna bodov získavajú tzv. **kľúčové body**, pre ktoré sa následne počítajú deskriptory (viď kapitola 3.3).

Po vypočítaní nasleduje **párovanie deskriptorov** medzi modelom a scénou (viď kapitola 3.5). Vykonaním tohto kroku získame zoznam dvojíc bodov (medzi modelom a scénou), ktoré by mohli tvoriť zhodu medzi týmito mračnami. Tieto dvojice nazývame **korešpodencie** a pre nájdenie výsledných transformácií je nutné vykonať **zhlukovanie korešpondencií** (viď kapitola 3.6).

Keďže sa dátová štruktúra k-d strom využíva vo viacerých popísaných algoritmoch, je mu venovaná kapitola 3.7. Poslednú časť tejto kapitoly tvorí detailný popis analýzy hlavných komponentov (viď kapitola 3.8), ktorá je v tejto práci použitá k redukcii dimenzií niektorých deskriptorov.

## 3.1 Mračno bodov

Množinu bodov, ktorá sa nachádza v súradnicovom systéme, označujeme ako **mračno bodov** [1]. V trojrozmernom (priestorovom) súradnicovom systéme sú tieto body definované súradnicami X, Y, Z reprezentujúcimi vonkajší povrch objektu. Súradnicu Z získavame z hĺbkovej mapy. Často sa k týmto súradniciam pridáva informácia o farbe, reprezentovaná modelom RGB [2]. Mračno bodov môžeme vytvoriť pomocou 3D senzorov alebo skenerov.

**Organizované mračno bodov** je označenie mračna, ktoré sa štruktúrou podobá matici, kde sú dáta rozdelené do riadkov a stĺpcov. Takéto mračná bodov pochádzajú zo stereo alebo Time of Flight (ToF) kamier. Výhodou je informácia o vzťahu medzi susednými bodmi (napr. pixelmi) - tým pádom sú metódy používajúce vyhľadávanie blízkych susedov jednoduchšie a rýchlejšie.

**Neorganizované mračno bodov** tvorí neusporiadanú množinu bodov. Nastáva tu problém s určením najbližšieho bodu, alebo okolia bodu. Možnosti určenia sú detailnejšie popísané v kapitole 3.1.2.

Tieto zariadenia merajú veľké množstvo bodov na povrchu objektu, ktoré následne zapisujú na výstup, väčšinou do súboru. Takto vytvorené mračná bodov sa dajú priamo vykresľovať a spracovávať, avšak pre použitie v niektorých 3D aplikáciach je potrebné vykonať **rekonštrukciu povrchu**.

Rekonštrukcia povrchu je proces, pri ktorom konvertujeme mračno bodov do trojuholníkovej (všeobecne mnohouholníkovej) siete. Často používaný algoritmus pre trianguláciu množiny bodov je Delaunayho triangulácia [5].

### 3.1.1 Reprezentácia mračna bodov

V súčasnej dobe existuje niekoľko rôznych formátov ako reprezentovať v počítači mračná bodov získané z laserových skenerov. Medzi najznámejšie patria: PLY, STL, OBJ a X3D.

Všetky tieto formáty súborov majú niekoľko nedostatkov, keďže boli vytvorené pre iné účely v dobe, kedy ešte neboli snímacie zariadenia dostupné na takej úrovni ako dnes. Preto bol autormi knižnice Point Cloud Library (skrátene **PCL**) [13], ktorá slúži na spracovanie 2D/3D obrázkov a mračien bodov (viď kapitola 4.1), vyvinutý nový formát súborov **PCD** (Point Cloud Data) [3]. Oproti starším formátom predstavuje niekoľko vylepšení:

- možnosť uchovávať a spracovávať organizované mračno bodov - dôležité pre real-time aplikácie a výskumné oblasti, ako rozšírená realita, robotika, atď.
- možnosť uchovávať hodnoty v rôznych dátových typoch (podporované všetky primitívne dátové typy: char, short, int, float, double) - použitie robí mračno bodov flexibilným a efektívnym pri ukladaní a spracovávaní
- možnosť ukladania N-dimenzionálnych histogramov pre deskriptory - dôležité pri aplikáciach v oblasti počítačového videnia

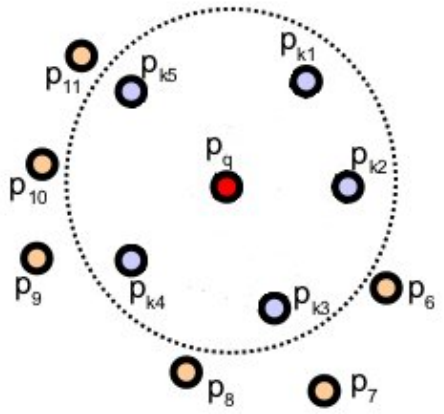
### 3.1.2 Okolie bodu

Ako bolo spomenuté v kapitole 3.1, mračno bodov môže byť organizované alebo neorganizované. V praxi sa väčšinou stretávame s druhým prípadom, takže nastáva problém s určením najbližšieho bodu alebo okolia bodu. Táto informácia je v praxi dôležitá pre rôzne metódy. Jedna z najdôležitejších je výpočet normál pre popísanie povrchu.

Na určenie okolia bodu $p_q$ sa v praxi často využíva dátová štruktúra k-d strom (viď kapitola 3.7). Väčšinou hľadáme okolie bodu dvoma spôsobmi:

1. zadaním hodnoty $d$, ktorá reprezentuje polomer abstraktnej gule v 3D priestore (v nej sa naše body musia nachádzať, je to teda ohraničujúci priestor),

2. zadaním hodnoty $k$, ktorá reprezentuje počet susedov s najmenšou priestorovou vzdialenosťou od bodu $p_q$




Obr. 1: Reprezentácia susedných bodov bodu $p_q$ v 2D priestore, resp. vyčlenenie bodov, ktoré patria do okolia bodu $p_q$ a ktoré nepatria.

### 3.1.3 Popis povrchu objektu v mračne bodov

K popisu povrchu objektov je možné použiť niekoľko metód. Jednou z jednoduchších a používanejších metód je odhad normál v jednotlivých bodoch mračna. Preto je im v tejto práci venovaná celá kapitola 3.2 .

## 3.2 Normály

**Normála** je v geometrii priamka, ktorá je kolmá na daný podpriestor. Vektor určujúci smer normály je **normálový vektor**. V priestorovom prípade je to vektor, ktorý je kolmý na rovinu. Normály sú dôležité vlastnosti geometrického povrchu a sú často využívané v rôznych aplikáciách počítačovej grafiky.

Ak máme daný geometrický povrch, je väčšinou jednoduché si odvodiť smer normály v danom bode na povrchu - ako vektor, ktorý je kolmý na povrch v tom bode. V prípade, že pracujeme s mračnami bodov, tak v dôsledku toho, že sú body reprezentované na reálnom povrchu, máme dve možnosti:

- získať povrch z daného mračna bodov, pomocou rekonštrukcii povrchu. Normály potom vypočítame z rekonštruovaného povrchu
- použiť aproximačné metódy k odvodeniu normál z mračna bodov priamo

V praxi je výhodnejšie použitie druhej možnosti, keďže rekonštrukcia povrchu vyžaduje čas, ktorý sa snažíme minimalizovať.

Existuje niekoľko spôsobov ako vykonať odhad normál z mračna bodov priamo. Jeden z najjednoduchších spôsobov je aproximácia problému odhadu normály v mračne bodov z rovinnej dotyčnice k ploche. Plocha má výsledný tvar najmenšieho štvorca vhodnej plochy pre daný odhad [12]. Plocha je daná bodom $x$ a vektorom $\vec{n}$. Postup odhadu normály pre bod $p_i$, ktorý sa nachádza v mračne bodov, je nasledovný:

- definujeme vzorec pre vzdialenosť bodu $p_i$ od plochy ako

$$d_i = (p_i - x) \cdot \vec{n} \tag{1}$$

- následne zadefinujeme výpočet centroidu mračna bodov

$$x = \overline{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \tag{2}$$

- položením $d_i = 0$ zistíme, že riešenie pre $\vec{n}$ je dané analýzou vlastných čísel a vektorov kovariančnej matice $\mathcal{C} \in \mathbb{R}^{3\times3}$, vyjadrenej ako:

$$\mathcal{C} = \frac{1}{k} \sum_{i=1}^{k} \cdot (p_i - \overline{p}) \cdot (p_i - \overline{p})^T, \ \mathcal{C} \cdot \vec{\mathsf{v}_j} = \lambda_j \cdot \vec{\mathsf{v}_j}, \ j \in \{0, 1, 2\} \tag{3}$$

kde $k$ je počet susedov, ktorý sa počíta do okolia bodu $p_i$, $\overline{p}$ reprezentuje 3D centroid najbližších susedov vypočítaný vzorcom 2, $\lambda_j$ je j-té vlastné číslo kovariančnej matice a $\vec{\mathsf{v}_j}$ je j-tý vlastný vektor. Aproximáciou $+\vec{n} = \{n_x, n_y, n_z\}$ (alebo $-\vec{n}$) je vlastný vektor $\vec{v_j}$ s najmenším vlastným číslom $\lambda_j$.

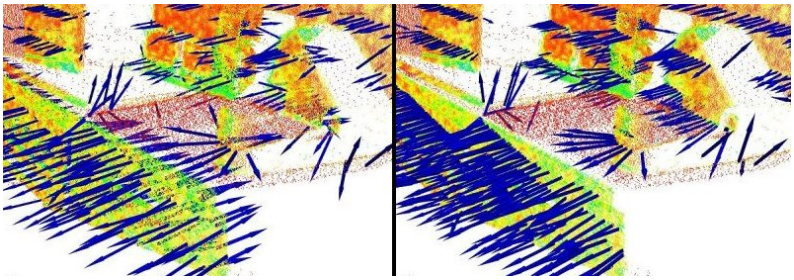
Analýzu vlastných čísel a vektorov nazývame analýzou hlavných komponentov (Principal Component Analysis, PCA) [4].

Problém, ktorý nastáva je, že nevieme určiť orientáciu normály. Získaná orientácia pomocou PCA analýzy je nejasná a nie je konzistentne orientovaná v celom mračne bodov. Obrázok 2 ukazuje orientáciu normál pred a po nasmerovaní k bodu kamery.

Riešenie tohto problému je triviálne, pokiaľ poznáme bod kamery, z ktorého bola scéna snímaná. K zorientovaní všetkých normál $\vec{n_i}$ konzistentne k bodu kamery musia všetky spĺňať podmienku:

$$\vec{n_i} \cdot (\mathsf{v}_p - p_i) > 0 \tag{4}$$

Dôležitým parametrom je určenie rozumného okolia bodu ($k$-okolia), ktoré sa bude podieľať na výpočte normál. Toto okolie by malo zodpovedať scéne, pre ktorú chceme odhadovať normály. Ak napríklad počítame normály pre model hrnčeku, kde je dôležité zakrivenie medzi uchom hrnčeka a hrnčekom, okolie by malo byť dostatočne malé na zachytenie týchto informácií a opačne.



Obr. 2: Na obrázku vľavo je vidieť nekonzistentne orientované normály $\vec{n_i}$. Vpravo je výsledok po preorientovaní normál konzistentne k bodu kamery.

### 3.3 Kľúčové body

Senzory ako napríklad Kinect alebo Xtion produkujú mračno s 307200 (640x480) bodmi. V praxi to znamená dosť veľké číslo pre spracovanie. Ak by sme chceli spraviť jednoduchú operáciu pre každý bod v mračne, tak by bola časová zložitosť $O(n)$, kde $n$ je počet bodov v mračne. Ak by sme mali porovnávať každý bod s jeho $k$ najbližšími susedmi, tak by bola časová zložitosť $O(nk)$. Pri plnom rozlíšení by sme stratili zbytočne veľa času na spracovávaní bodov, ktoré ani nepotrebujeme.

Preto sa využívajú rôzne optimalizačné techniky, ako napríklad paralelizácia pomocou GPU. Druhou, častejšie využívanou technikou, je zníženie zložitosti mračna bodov odstránením všetkých bodov, ktoré nie sú potrebné. Tento proces vedie k získaniu tzv. **kľúčových bodov**.

Kľúčové body (označované ako body záujmu) sú body v obrázku, alebo v mračne bodov, ktoré sú stabilné, charakteristické a môžu byť identifikované pomocou dobre definovaného detekčného kritéria. Počet týchto bodov bude typicky v mračne bodov oveľa menší ako celkový počet bodov. V praxi máme dva spôsoby získavania takýchto bodov:

- podvzorkovanie mračna bodov
- využiť algoritmy určené k detekcii kľúčových bodov

### 3.3.1 Podvzorkovanie

Výsledkom podvzorkovania je mračno bodov, ktoré obsahuje podstatne menej bodov ako pôvodné mračno, avšak pri zachovaní pôvodného tvaru a presnosti pre spracovanie.

Tento proces je vykonaný pomocou voxelizácie. Voxel je v podstate 3D verzia pixelu. V prvom kroku je mračno rozdelené do niekoľkých oblastí v tvare kocky s požadovaným rozlíšením. Následne sú všetky body v každom voxele spracované tak, že v každom zostane iba jeden bod.

Najjednoduchšie spracovanie voxelu by pozostávalo z náhodného výberu jedného bodu, avšak presnejšou metódou je vypočítať centroid, čiže bod, ktorého súradnice sú stredné hodnoty všetkých bodov patriacich do voxelu.

Pri správne zvolenom parametri vráti podvzorkovanie výsledky, ktoré sú dostatočne presné k spracovaniu, za cenu ušetrenia výpočtovej kapacity. Tento parameter sa v praxi nazýva aj ako veľkosť listu (viď príloha A.2, parametre `model_ss_` a `scene_ss_`).

### 3.3.2 Algoritmy detekcie kľúčových bodov

Existuje niekoľko používaných detektorov kľúčových bodov. Niektoré z nich majú základ vo svojich 2D verziách. Najznámejší z nich je Scale Invariant Feature Transform (SIFT), ktorý je možné rozdeliť na štyri kroky:

- konštrukcia scale-space
- výpočet rozdielových obrázkov
- odstránenie kľúčových bodov s nízkym kontrastom
- priradenie orientácie kľúčovým bodom

V prvom kroku sú vytvorené vnútorné reprezentácie originálneho obrazu, aby sa zachovala odolnosť voči zmenám mierky. Dosiahne sa tým tak, že sa vytvorí tzv. scale-space, čo je vlastne vygenerovanie pyramídy, ktorá pozostáva z niekoľkých oktáv pôvodného obrázku s inými rozlíšeniami (a postupným Gaussovským rozostrením).

Po vygenerovaní pyramídy sú vypočítané rozdielové obrazy dvoch susedných gausiánov v každej oktáve - Difference of Gaussians (DoG).

Následne sú v rozdielových obrazoch nájdené lokálne extrémy, ktoré sú ďalej vyšetrené tak, aby body s nízkym kontrastom a body ležiace na hranách boli odstránené.

Nakoniec je potrebné pridať orientáciu kľúčovým bodom, vďaka čomu dosiahneme odolnosť voči rotácii.

Ďalšie používané algoritmy sú Intrinsic Shape Signatures (ISS) a Harris3D.

## 3.4 Deskriptory

V kapitole 3.2 boli predstavené povrchové normály, ktoré popisujú povrchové okolie bodu (vďaka výpočtu, do ktorého sa zahrňuje $k$ zvolených najbližších susedov, alebo všetci najbližší susedia v polomere $r$). Normály sú základné deskriptory, avšak nie optimálne a v praxi ich nosná informácia nestačí k rozpoznaniu objektu v obraze.

Všeobecne sú deskriptory zložitejšie (a presnejšie) nosiče viacerých informácií o geometrii okolia bodu, pre ktorý boli vypočítané. K tomu, aby boli deskriptory optimálne, musia spĺňať nasledujúce kritéria:

- odolnosť voči transformáciám - konkrétne k hmotným transformáciám (nemenia vzdialenosti medzi bodmi), takže napríklad rotácia a posuv by nemali mať vplyv na deskriptor
- odolnosť voči šumu - chyby merania, ktoré spôsobujú šum, by nemali mať veľký vplyv na odhad deskriptorov
- nemennosť pri zmene mierky - keď sa vykoná vzorkovanie s inou hustotou mračna (napríklad po podvzorkovaní), výsledky musia byť rovnaké, alebo podobné

Hlavnou myšlienkou je jednoznačne identifikovať bod naprieč viacerými mračnami bodov, nezávisle na šume, rozlíšení alebo transformácii. Niektoré z deskriptorov zachytávajú aj informácie o objekte, do ktorého patria, ako napríklad bod kamery, čo v konečnom dôsledku slúži k získaniu pozície.

### 3.4.1 Lokálne deskriptory

Metódy založené na lokálnych deskriptoroch popisujú iba povrchy v okolí zvolených kľúčových bodov. Na rozdiel od metód založených na globálnych deskriptoroch majú lepšie výsledky rozpoznávania v preplnených scénach. Tento typ deskriptorov sa ukázal ako lepší aj pri rozpoznávaní v 2D obraze. Podobný záver sa neskôr rozšíril aj do rozpoznávania v 3D obraze, preto sa tieto metódy používajú najčastejšie.

Normály (viac v kapitole 3.2) väčšinou tvoria základ k popisu geometrie okolo bodu. Výpočet je pomerne jednoduchý, ale vzhľadom k tomu, že aproximujú geometriu bodov málo hodnotami, je pravdepodobné, že väčšina scén bude mať niekoľko bodov, ktoré budú mať podobné hodnoty deskriptorov. Týmto sa znižuje ich schopnosť dobre popisovať povrch okolo bodu. Normály sa využívajú ako základ pre viacero lokálnych deskriptorov v kombinácii s inými vlastnosťami, aby sa dosiahlo lepšej popisnej vlastnosti okolia bodu. V súčastnosti dosahujú najlepšie výsledky tieto deskriptory:

- Signatures of Histograms of OrienTations (SHOT)
- Point Feature Histogram (PFH)
- Fast Point Feature Histogram (FPFH)

Postup, ktorý je potrebné vykonať pri použití tohto druhu deskriptorov je zobrazený na obrázku 3. Získaním kľúčových bodov (viac v kapitole 3.3) a počítaním deskriptorov iba v týchto bodoch markantne zrýchlime celý proces. Následne vykonáme hľadanie dvojíc bodov, ktoré by mohli tvoriť zhodu medzi scénou a modelom. Proces prebieha pomocou zvoleného vyhľadávacieho algoritmu založeného na hľadaní najbližších bodov - najčastejšie k-d strom (viac v kapitole 3.7). Následne sa vykoná zhlukovanie korešpondencií (viac v kapitole 3.6), z ktorých nájdeme možné transformačné matice inštancií modelov nájdených v scéne. V tomto kroku buď zoberieme transformáciu, do ktorej "padlo"najviac korešpondencií (alebo N najlepších transformácií), alebo vykonáme ďalšie kroky následného spracovania na spresnenie transformácií a vyradenie nesprávnych transformácií.


Obr. 3: Postup pri odhade 6D pozície objektu pomocou lokálnych deskriptorov

Keďže sa práca zameriava na použitie a rozšírenie deskriptoru FPFH, ktorého základ tvorí deskriptor PFH, sú tieto dva algoritmy popísané v nasledujúcich kapitolách.

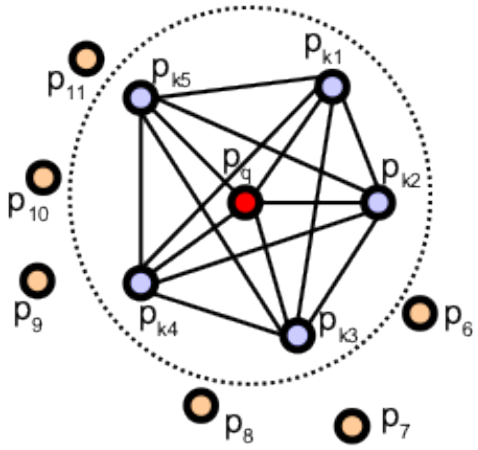
#### 3.4.1.1 Point Feature Histogram (PFH)

Metóda využíva pre výpočet výsledného popisu odhady povrchových normál a zakrivení geometrických tvarov v okolí počítaného bodu $p_q$. Obrázok 4 predstavuje vzťahy medzi dvojicami bodov v okolí $p_q$ (resp. oblasť, z ktorej sa počíta deskriptor v danom bode $p_q$). Ako je vidieť, dvojice bodov tvoria plne prepojenú sieť.

Výsledný PFH deskriptor je vytvorený ako M-dimenzionálny histogram hodnôt, ktorý reprezentuje geometrické vlastnosti a odhad normál v okolí bodu medzi všetkými dvojicami bodov, ktoré do okolia patria. Inak povedané, metóda sa snaží zachytiť čo najlepšie povrchové zmeny zo vzorku tak, že zohľadňuje všetky interakcie medzi smermi odhadnutých normál povrchu. Výsledný hyper priestor je preto veľmi závislý na kvalite odhadnutých normál.

Časová náročnosť pre výpočet PFH deskriptoru je $O(k^2)$, kde $k$ je počet susedov, ktorí sa na výpočte podieľajú. Postup pre výpočet deskriptoru je nasledovný:

- v prvom kroku je potrebné odhadnúť normály povrchu $\vec{n_i}$ pre všetky body z okolia bodu (väčšinou sa ako časť predspracovania odhadnú normály z celého mračna bodov),




Obr. 4: Vzťahy dvojíc bodov v okolí bodu $p_q$, pre ktoré sa počíta deskriptor.

- pre výpočet relatívneho rozdielu medzi bodmi $p_s$ a $p_t$ a ich normálami $\vec{n_s}$ a $\vec{n_t}$ definujeme fixný Darbouxov rámec (fixné súradnice vektorov $\vec{u}$, $\vec{v}$ a $\vec{w}$) pre jeden z dvoch bodov. Ak označíme $p_s$ ako zdrojový bod a $p_t$ ako cieľový bod, za $p_s$ dosadíme ten bod, ktorý zviera najmenší uhol medzi svojou normálou a čiarou spájajúcu body $p_s$ a $p_t$ (viď obrázok 5). Základ Darbouxového rámca môžeme v bode $p_s$ definovať ako:

$$\begin{aligned}\vec{u} &= \vec{n_s} \\ \vec{v} &= \vec{u} \times \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ \vec{w} &= \vec{u} \times \vec{v}\end{aligned}$$

- použitím Darbouxového rámca **uvw** definujeme rozdiel dvoch normál $\vec{n_s}$ a $\vec{n_t}$ ako množinu uhlových deskriptorov následovne:

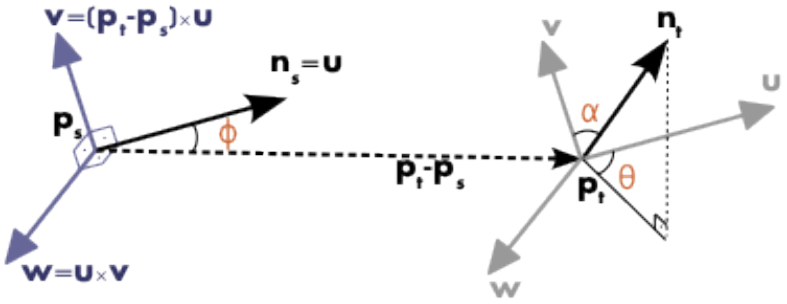
$$\begin{aligned}\alpha &= \vec{v} \cdot \vec{n_t} \\ \phi &= \vec{u} \cdot \frac{(p_t - p_s)}{d} \\ \theta &= \arctan(\vec{w} \cdot \vec{n_t}, \vec{u} \cdot \vec{n_t})\end{aligned}$$

kde $d$ je Euklidovská vzdialenosť medzi dvomi bodmi $p_s$ a $p_t$, $d = \|p_t - p_s\|_2$. K týmto trom uhlom je pridaná vzdialenosť $d$. Je potrebné však dodať, že táto hodnota deskriptoru nie je v niektorých scénach významná [8]. V praxi sa aj napriek tomu štvorica $\langle \alpha, \phi, \theta, d \rangle$ počíta pre

každú dvojicu bodov v okolí bodu, pre ktorý deskriptor odhadujeme. Vo výsledku dosiahneme redukciu 12 hodnôt (súradnice X, Y, Z a normálové informácie pre dva body) na 4.

Tieto štvorice sa využívajú k zostaveniu výslednej reprezentácie deskriptoru PFH pre daný bod. Jednotlivé hodnoty z danej štvorice sú rozdelené na $b$ pod-intervalov a počíta sa hodnota výskytov bodov v týchto intervaloch. Keďže sú 3 z hodnôt v tejto štvorici uhly a sú potrebné pre popis povrchu, je možné tieto hodnoty jednoducho normalizovať do rovnakého intervalu na jednotkovej kružnici.

Tieto normalizované hodnoty sú následne počítané do intervalu tvoriaci výsledný histogram. Časová zložitosť algoritmu je $O(nk^2)$.




Obr. 5: Grafická reprezentácia Darbouxového rámca a uhlových PFH deskriptorov dvojice bodov $p_s$ a $p_t$.

#### 3.4.1.2 Fast Point Feature Histogram (FPFH)

Ako bolo vysvetlené v kapitole 3.4.1.1, teoretická časová náročnosť pre výpočet deskriptorov PFH v mračne bodov $P$ s $n$ bodmi je $O(nk^2)$, kde $k$ je počet susedov pre každý bod $p$ v mračne bodov $P$. V aplikáciach, ktoré musia pracovať v reálnom čase, môže výpočet deskriptorov Point Feature Histogram v hustom mračne bodov predstavovať jeden z hlavných úskalí.

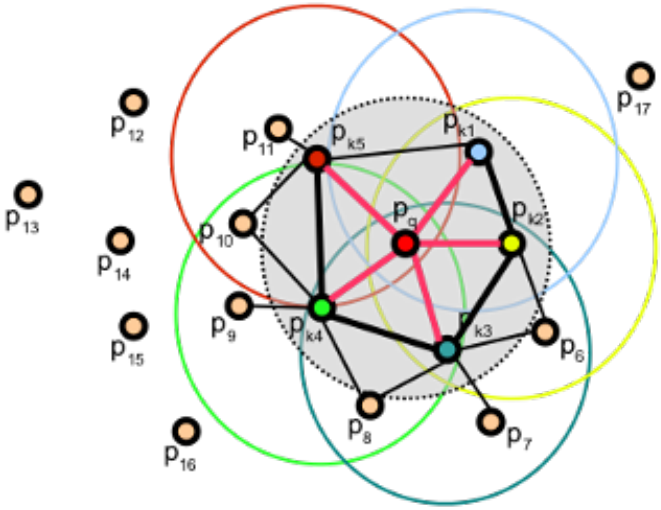
Táto kapitola popisuje zjednodušenie formulácie PFH deskriptorov, ktoré sa nazýva Fast Point Feature Histogram (FPFH) [9]. Teoretická časová zložitosť algoritmu je $O(nk)$ pri zachovaní popisných vlastnosti PFH. Pre zjednodušenie výpočtu postupujeme nasledovne:

- v prvom kroku sa pre každý bod $p_q$ v mračne bodov vypočíta trojica uhlových hodnôt $\langle \alpha, \phi, \theta \rangle$ medzi ním a jeho susednými bodmi, rovnakým spôsobom, akým je popísané v kapitole o PFH (kapitola 3.4.1.1). Tento výpočet sa nazýva Simplified Point Pair Feature (SPFH)
- v druhom kroku je pre každý bod $p_q$ vypočítaná hodnota FPFH deskriptoru za pomoci SPFH hodnôt bodu $p_q$ a susedných bodov podľa vzorca:

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{\omega_k} \cdot SPFH(p_k) \tag{5}$$

kde váha $\omega_k$ reprezentuje vzdialenosť medzi bodom $p_q$ a susedným bodom $p_k$ v zvolenej metrike.

Výsledné hodnoty sú závislé na hodnotách susedných bodov a tie zase na hodnotách vo svojom okolí. Obrázok 6 zobrazuje vzťah medzi bodom $p_q$ a jeho susednými bodmi (vďaka vypočítaným SPFH hodnotám) vyznačený červenými čiarami. Dodatočné spojenia získané vďaka vylepšeniu PFH deskriptorov sú vyznačené čiernymi čiarami. Niektoré páry sa počítajú dvakrát, na obrázku sú vyznačené čiernymi hrubšími čiarami.




Obr. 6: Zobrazenie všetkých vzťahov medzi bodom $p_q$ a jeho susednými bodmi v metóde FPFH.

Aby bolo možné odhadovať FPFH deskriptory v reálnom čase, algoritmus sa líši od formulácie PFH v nasledovných bodoch:

- FPFH plne neprepája všetkých susedov bodu a tým pádom mu chýbajú niektoré páry, ktoré môžu prispieť k presnejšiemu popisu povrchu okolo daného bodu
- PFH modeluje presne daný povrch v okolí bodu $p_q$, na rozdiel od toho FPFH zahrňuje ďalšie body mimo polomeru $r$ (avšak najviac v okolí $2r$)
- kvôli prehodnocovaniu hodnôt FPFH kombinuje SPFH hodnoty a znova získava niektoré susedné páry
- výsledný histogram je zjednodušený odstránením súvzťažnosti medzi jednotlivými hodnotami tak, že sa vytvorí $d$ rôznych histogramov, jeden pre každú dimenziu deskriptoru a následným spojením dohromady

### 3.4.2 Globálne deskriptory

Metódy založené na globálnych deskriptoroch spracovávajú rozpoznávaný objekt ako celok. Definujú niekoľko deskriptorov, ktoré efektívne a stručne popisujú celý 3D objekt (alebo model). Tieto metódy sú používané v súvislosti so získavaním 3D tvaru a klasifikáciou a patria medzi nich:

- geometrické 3D momenty [7]
- Viewpoint Feature Histogram [10]

Avšak tieto metódy ignorujú tvarové detaily a potrebujú segmentáciu objektu zo scény ako časť predspracovania. Z tohto dôvodu sa nevyužívajú pre rozpoznanie čiastočne viditeľných objektov v preplnenej scény. V prípade použitia je potrebné postupovať krokmi znázornenými na obrázku 7 . V prvom kroku je nutné vykonať segmentáciu mračna bodov, vďaka ktorej získame všetky možné objekty, reprezentované zhlukmi. Pre každý zhluk sa potom vypočíta globálny deskriptor a hľadajú sa korešpondencie podobne ako v prípade použitia lokálnych deskriptorov.

Výsledná pozícia môže byť získaná vypočítaním a nasmerovaním centroidov zhlukov. Ďalej pokračujeme následným spracovaním ako v prípade lokálnych deskriptorov.


Obr. 7: Postup pri odhade 6D pozície objektu pomocou globálnych deskriptorov

Z dôvodu názornej ukážky rozdielov oproti lokálnym deskriptorom je v nasledujúcej kapitole uvedený detailnejší popis výpočtu deskriptoru Viewpoint Feature Histogram.

#### 3.4.2.1 Viewpoint Feature Histogram (VFH)

Táto metóda má základy v deskriptore FPFH (viď kapitola 3.4.1.2). Vzhľadom k jeho rýchlosti a rozpoznávacej sile sa autori rozhodli využiť výpočty relatívnych uhlov normál v kombinácii so smerom pohľadu kamery senzoru $V_p$ (pozri obrázok 8).

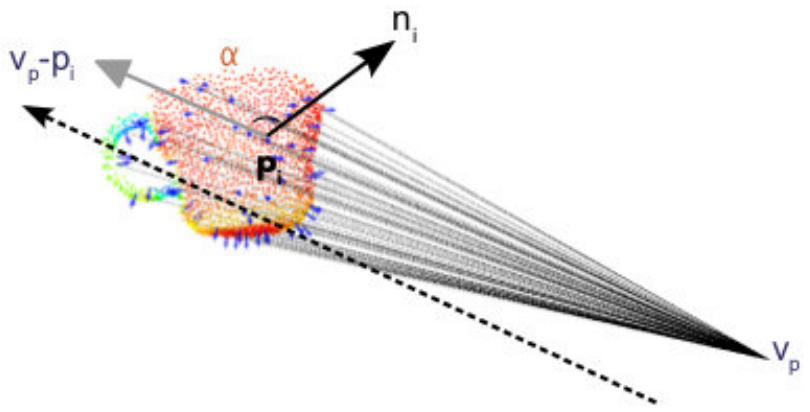
Výhodou je predovšetkým rýchlosť výpočtu, ktorá je takmer v reálnom čase. Časová zložitosť algoritmu je $O(n)$, kde n je počet bodov v mračne. VFH si zachováva veľkú rozpoznávaciu silu, akou disponuje FPFH a je odolný voči zmene mierky objektov.

VFH deskriptor pozostáva z dvoch častí:

1. komponent obsahujúci smer pohľadu kamery,
2. komponent obsahujúci tvar povrchu získaný z rozšíreného výpočtu FPFH.

Výpočet komponentu, ktorý obsahuje smer pohľadu kamery $V_p$ prebieha tak, že sa získa histogram uhlov, ktoré zviera vektor pohľadu kamery $V_p$ s normálou v každom bode mračna. Je potrebné dodať, že sa nejedná o uhol pohľadu s každou normálou, pretože by takýto výpočet nebol odolný voči zmenám mierky. Namiesto toho sa používa centrálny smer $V_p$ a uhly zvierajúce so všetkými normálami. Nejedná sa teda o viditeľné uhly normál, pretože by potom výsledky neboli odolné voči rôznym mierkam objektov. Objekt by potom musel byť neustále v rovnakej vzdialenosti od kamery.

Druhý komponent obsahuje relatívne hodnoty uhlov, počítané rovnako ako v algoritme FPFH (viac v kapitole 3.4.1.2) s tým rozdielom, že sa počítajú voči $V_p$.




Obr. 8: Smer pohľadu kamery a komponenty pre výpočet uhlov vo VFH

## 3.5 Párovanie deskriptorov

Po vypočítaní lokálnych 3D deskriptorov pre scénu nasleduje hľadanie možných dvojíc korešpondenčných bodov, ktoré by mohli predstavovať správnu transformáciu modelu v scéne.

Musíme pre každý kľúčový bod v mračne bodov scény vykonať párovanie deskriptorov hľadaním zhôd s deskriptormi vypočítanými pre model (napr. v databáze modelov). Pre tento účel sa využívajú štruktúry ako k-d strom (viď kapitola 3.7), pomocou ktorých sa vykonáva vyhľadávanie najbližších susedov získavaním Euklidovskej vzdialenosti medzi deskriptormi a hľadaním dvojíc bodov s najmenšími $k$ vzdialenosťami. Parameter $k$ je definovaný ako vstup, väčšinou sa používa $k > 1$ a ovplyvňuje výsledný počet korešpondencií (čím väčšie je $k$, tým vyšší bude počet korešpondencií).

Každý deskriptor zo scény musí byť porovnaný so všetkými deskriptormi modelov v databáze, keďže sa v scéne môže nachádzať niekoľko inštancií modelov.

Vyššie popísaný postup sa dá spresniť tak, že vylúčime niektoré korešpondencie napríklad na základe maximálnej vzdialenosti medzi bodmi (zadaním prahovej hodnoty).

## 3.6 Zhlukovanie korešpondencií

Výsledkom párovania deskriptorov, popísaného v kapitole 3.5, máme k dispozícii zoznam korešpondencií medzi kľúčovými bodmi v scéne a niektorými modelmi z databáze. Avšak táto informácia nám ešte nezaručuje, že daný objekt sa v scéne nachádza.

Zoberme si príklad, kedy máme krabicu s dvomi kľúčovými bodmi umiestnenými v opačných rohoch. Vzdialenosť medzi bodmi je známa. Pri párovaní deskriptorov sa našli dva kľúčové body v scéne s veľmi podobnými popismi, ale ako sa ukazuje, vzdialenosť medzi nimi je veľmi odlišná. Pokiaľ neberieme do úvahy transformácie ako napríklad úprava mierky, tak zistíme, že tieto body nepatria do objektu.

Táto kontrola je implementovaná v kroku, ktorý nazývame **zhlukovanie korešpondencií**. Ako naznačuje názov, metóda zoskupí korešpondencie, ktoré sú geometricky konzistentné (pre daný model objektu) a ostatné vyradí.

Rotácie a posuvy sú povolené, ale iné transformácie nesplnia kritéria. Musíme tiež počítať s tým, že pre získanie 6 DoF pozície objektu je potreba minimum troch korešpondencií. Zhluky s menším počtom korešpondencií budú ignorované.

### 3.6.1 Geometrická konzistencia

Jedná sa o jednu z najľahších metód. Algoritmus iteratívne prechádza všetky korešpondencie, ktoré ešte neboli zaradené a pridáva ich do práve spracovávanej podmnožiny, pokiaľ sú geometricky konzistentné [19].

Každá podmnožina predstavuje zoznam korešpondencií, ktoré patria do jednej špecifickej transformácii modelu v scéne.

V prvom kroku sa teda zvolí počiatočná korešpondencia $c_i = ({p_i}^m, {p_i}^s)$ (kde ${p_i}^m$ predstavuje kľúčový bod z modelu a ${p_i}^s$ zo scény) a postupne sa prechádzajú všetky nezaradené korešpondencie. Korešpondencia $c_j = ({p_j}^m, {p_j}^s)$ bude pridaná do podmnožiny vytvorenej z korešpondencie $c_i$ ak bude splnená nasledujúca podmienka:

$$|\|{p_i}^m - {p_j}^m\|_2 - \|{p_i}^s - {p_j}^s\|_2| < \epsilon \tag{6}$$

kde $\epsilon$ je parameter tejto metódy (viď príloha A.2, parameter `cg_size`), ktorý ovplyvňuje, ako moc bude vynútená geometrická konzistencia medzi korešpondenciami. Ďalší parameter metódy je prahová hodnota, definujúca najmenší počet korešpondencií, ktoré môže podmnožina obsahovať (viď príloha A.2, parameter `cg_thresh`). Podmnožiny s menším počtom budú vylúčené. Minimálny počet pre získanie 6 DoF pozície objektu je 3, ale v praxi sa definuje toto číslo väčšie, napr. 5 až 8.

### 3.6.2 Houghové hlasovanie

Táto metóda používa techniku, ktorá sa nazýva Houghova transformácia [20]. Technika bola pôvodne navrhnutá k detekcii čiar v 2D obraze, ale bola neskôr rozšírená tak, aby dokázala pracovať s ľubovoľnými tvarmi alebo vyššími dimenziami.

Metóda pracuje so systémom hlasovania, v ktorom sú hlasy odovzdávané kandidátom, ktorí sa ukážu ako vhodnejší. Keď sa získa dosť hlasov pre danú transformáciu modelu v scéne, je táto transformácia považovaná ako tá správna.

Hlasovanie sa vykonáva v priestore, ktorý by mal mať 6 dimenzií v prípade, že chceme získať celkovú pozíciu objektu (rotácia a posuv). V praxi by ale bola táto procedúra výpočtovo náročná, preto sa často používa 3D Houghov priestor a využívajú sa lokálne referenčné rámce na získanie zostávajúcich 3 stupňov voľnosti.

## 3.7 K-d strom

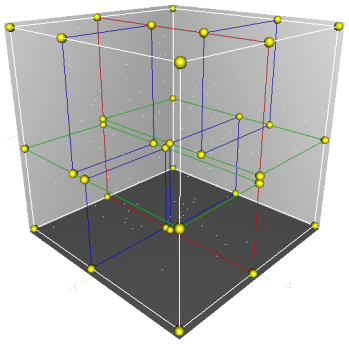
Táto dátová štruktúra organizuje množinu bodov v k-dimenzionálnom priestore takým spôsobom, aby boli operácie vyhľadávania v okolí veľmi efektívne. V praxi sa väčšinou využíva na dve operácie:

- vyhľadávanie najbližších $n$ susedných bodov v $k$ dimenzionálnom priestore
- nájdenie všetkých susedných bodov v danom polomere $r$

Jedná sa o druh binárneho stromu, takže každý uzol, ktorý nie je list, má dvoch potomkov (ľavého a pravého). Každá úroveň rozdeľuje priestor na špecifickú dimenziu.

Napríklad v 3-dimenzionálnom priestore by v koreňovom uzli boli všetci potomkovia rozdelení na základe prvej dimenzie, takže na X súradnice (body s väčšou X súradnicou by boli v ľavom pod-strome, menšie v pravom). Na druhej úrovni (uzly, ktoré sme práve vytvorili), rozdelenie by bolo vykonané na základe Y súradnice s rovnakými kritériami. Na tretej úrovni by bola použitá Z súradnica. Na štvrtej úrovni by sme sa vrátili naspäť na X os a tak ďalej.

Väčšinou sa používa stredový bod ako koreň na každej úrovni. Nové úrovne sa tvoria až pokým posledné pod-stromy, ktoré chceme rozdeliť, nebudú jednoprvkové.



Obr. 9: 3-dimenzionálny k-d strom.

## 3.8 Analýza hlavných komponentov

Analýza hlavných komponentov (anglicky Principal Component Analysis, PCA) je matematická štatistická metóda, v ktorej hľadáme množinu lineárnych kombinácií pôvodných dát tak, aby sme zachovali čo najväčšie množstvo informácií o pôvodných dátach a zároveň jej dimenzia bude menšia alebo nanajvýš rovná ako dimenzia pôvodnej množiny.

Týmto postupom sa docieli to, že bude možné reprezentovať pôvodné vysoko-dimenzionálne dáta pomocou menšieho počtu dimenzií pri zachovaní určitého percenta pôvodnej informácie.

Počet hlavných komponentov je teda vždy menší alebo nanajvýš rovný pôvodnému počtu prvkov. Ortogonálna transformácia, ktorá sa pri tejto metóde používa, je definovaná tak, aby mal prvý hlavný komponent najväčší rozptyl spomedzi všetkých možných lineárnych kombinácii vektora pozorovaní. Analýza prebieha v nasledovných krokoch:

1. štandardizácia dát
2. výpočet kovariančnej matice
3. výpočet vlastných čísel a vektorov
4. výber bázových vektorov
5. projekcia dát do nového pod-priestoru
6. výber vhodnej redukovanej dimenzie L

Pri vykonávaní analýzy predpokladáme, že vstupné dáta sú uložené v matici $X$ o rozmeroch $n \times p$, kde $n$ je počet pozorovaní (napr. deskriptorov) a $p$ reprezentuje dimenziu dát (počet hodnôt v danom pozorovaní).

Cieľom analýzy je redukcia dát tak, aby bolo možné každé pozorovanie reprezentovať menším počtom hodnôt $L$, kde $L \leq p$.

### 3.8.1 Štandardizácia dát

Základ analýzy je odčítanie strednej hodnoty z každej dimenzie vstupných dát. Strednú hodnotu vypočítame ako priemer všetkých pozorovaní v dimenzii $x_i$, pre ktorú priemer počítame.

Pre zjednodušenie vypočítame vektor stredných hodnôt. Tento vektor sa bude skladať z $p$ hodnôt a formálne môžeme výpočet zapísať nasledovne:

$$u[j] = \frac{1}{n} \sum_{i=1}^{n} X[i, j] \tag{7}$$

Následne odčítame vektor $u$ z každého pozorovania (tj. z každého riadku v matici $X$). Formálne môžeme operáciu zapísať nasledovne:

$$B = X - \vec{h} \cdot \vec{u}^{\top} \tag{8}$$

kde $\vec{h}$ je stĺpcový vektor o rozmeroch $n \times 1$, obsahujúci samé 1. Výsledok tejto operácie uložíme do matice $B$, ktorá bude mať identické rozmery ako matica $X$. Táto matica bude použitá v ďalších krokoch analýzy.

### 3.8.2 Výpočet kovariančnej matice

Kovariančná matica je symetrická matica o rozmeroch $p \times p$, kde $p$ je počet dimenzií vstupných dát. Na pozícii $C_{ij}$ sa v matici nachádza kovariancia medzi prvkami $i$ a $j$ náhodného vektoru. Na diagonálach sa nachádza rozptyl jednotlivých prvkov náhodného vektoru. Štruktúra matice je nasledovná:

$$C = \begin{bmatrix} \operatorname{var}(X_1) & \operatorname{cov}(X_1, X_2) & \cdots & \operatorname{cov}(X_1, X_p) \\ \operatorname{cov}(X_2, X_1) & \operatorname{var}(X_2) & \cdots & \operatorname{cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{cov}(X_p, X_1) & \operatorname{cov}(X_p, X_2) & \cdots & \operatorname{var}(X_p) \end{bmatrix} \tag{9}$$

Výpočet kovariančnej matice môžeme definovať nasledujúcim vzťahom (jedná sa o operácie medzi celými maticami, v praxi je implementácia takéhoto výpočtu v rámci niektorej z knižníc lineárnej algebry jednoduchšia a efektívnejšia):

$$C = \frac{1}{n-1} B^{\top} \cdot B \tag{10}$$

kde $n$ je počet pozorovaní (resp. počet riadkov v matici $B$), $B^T$ je transponovaná matica $B$. Delenie číslom $n-1$ sa používa kvôli tzv. Besselovej korekcii [18].

### 3.8.3 Výpočet vlastných čísel a vektorov

V tomto kroku sú z kovariančnej matice $C$ vypočítané vlastné čísla a vektory. Tento krok je v praxi väčšinou vykonaný pomocou knižníc určených pre výpočty lineárnej algebry, ako napríklad Eigen [14], ktorá počíta vlastné čísla a vektory pomocou rozkladu reálnej symetrickej matice $C$ na tvar:

$$C = VDV^{-1} \tag{11}$$

kde stĺpce ${s_i}^U$ matice $U$ sú vlastné vektory $C$ korešpondujúce vlastným číslam $\lambda_i$. Diagonálne prvky matice $D$ sú vlastné čísla $\lambda_i$ tak, že $i$-te vlastné číslo sa nachádza na pozícii $D_{ii}$. Všetky prvky v matici D nachádzajúce sa mimo diagonálu sú rovné 0.

Po získaní vlastných čísel a vektorov je potrebné ich zoradiť od najvyššieho vlastného čísla po najmenšie (s odpovedajúcim vlastným vektorom).

### 3.8.4 Výpočet súhrnnej energie pre každý vlastný vektor

Vlastné čísla reprezentujú distribúciu energie vstupných dát medzi každý z vlastných vektorov. Súhrnnú energiu $g$ pre $j$-tý vektor je suma všetkých energií vlastných čísel od 1 do $j$. Tento výpočet môžeme zapísať takto ako:

$$g[j] = \sum_{k=1}^{j} D[k,k] \qquad \text{pre} \qquad j = 1, \ldots, p \tag{12}$$

### 3.8.5 Výber bázových vektorov

Maticu bázových vektorov označujeme $W$ a bude použitá v ďalšom kroku pri projekcii dát. Zostavíme ju z vlastných vektorov vypočítaných z kovariančnej matici $C$ takým spôsobom, že zoberieme prvých $L$ vlastných vektorov, ktorým odpovedajú najvyššie vlastné čísla. Tým pádom bude mať $W$ rozmery $p \times L$. Zostavenie $W$ môžeme zapísať nasledovným spôsobom:

$$W[k,l] = V[k,l] \qquad \text{pre} \qquad k = 1, \ldots, p \qquad l = 1, \ldots, L \tag{13}$$

### 3.8.6 Projekcia dát do nového pod-priestoru

V tomto kroku transformujeme štandardizované dáta do nového priestoru, ktorý má $L$ dimenzií, kde $1 \leq L \leq p$. K tomu využijeme maticu $W$, ktorou vynásobíme maticu štandardizovaných dát následovne:

$$T = B \cdot W = \mathbb{KLT}\{X\} \tag{14}$$

kde $T$ je matica s redukovaným počtom dimenzií $L$ o rozmeroch $n \times L$ a riadky tejto matice reprezentujú Kosambi-Karhunen-Loéveho (KLT) transformáciu vektorov v riadkoch matice $X$.

### 3.8.7 Výber vhodnej redukovanej dimenzie L

Vektor súhrnných energií $g$ môžeme využiť k doporučeniu vhodného parametra $L$ pri redukcii dimenzií dát pomocou PCA analýzy.

Cieľom je zvoliť hodnotu $L$ čo najmenšiu pri zachovaní relatívne veľkého čísla $g$ na percentuálnej báze. Inými slovami, snažíme sa nájsť číslo $L$ tak, aby sme zachovali zvolené percento pôvodných informácií obsiahnutých vo vstupných dátach. V praxi volíme väčšinou hodnotu medzi 90-95%. Formálny zápis výpočtu je nasledovný:

$$\frac{g[L]}{g[p]} \geq t \tag{15}$$

kde $t$ je prahová hodnota v intervale $[0.0, 1.0]$.

# 4 Praktická časť

Cieľom praktickej časti je vytvorenie aplikácie na rozpoznávanie objektov v RGB-D obraze získaného zo senzoru Microsoft Kinect. Objekty sú reprezentované 3D modelmi, ktoré sú uložené ako mračno bodov vo formáte PCD. Scény, v ktorých sa hľadajú objekty, sú uložené v rovnakom formáte, ale obsahujú niekoľkonásobne viac bodov (napr. pozadie, viacero objektov, šum a pod.).

Výsledkom aplikácie je nájdenie čo najpresnejšej pozície objektu v 3D priestore. Pozícia je v praxi definovaná ako transformačná matica, resp. vektor posunu a matica rotácie.

Pri vývoji aplikácie je nevyhnutné vyskúšať rôzne State of the Art algoritmy pre tvorbu 3D deskriptorov a porovnať ich výsledky na dostupných dátových sadách. Zvolil som variantu výpočtu lokálnych deskriptorov, keďže v praxi tento spôsob dosahuje lepších výsledkov rozpoznávania (viď kapitola 3).

Praktická časť a experimenty v mojej práci sú zamerané na 3D deskriptory, pretože ich výpočet zaberá v celom procese rozpoznávania objektov skoro najviac času.

Aplikácia bola vyvinutá v programovacom jazyku C++ s použitím knižnice Point Cloud Library (PCL) [13], ktorá slúži na spracovanie 2D/3D obrazu a mračien bodov. Technické vybavenie počítača, na ktorom bola výsledná aplikácia testovaná, je nasledovné:

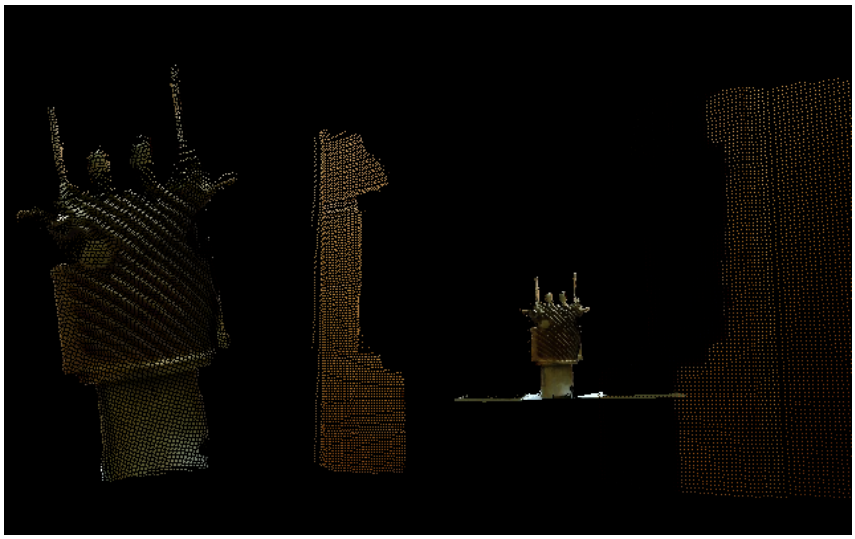
- Procesor: Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz
- Grafická karta: AMD Radeon HD 8570M
- Pamäť RAM: 6GB DDR3

Vývoj postupoval pridávaním krokov v podobnom poradí ako v kapitole 3 . Postupne budú popísané jednotlivé kroky:

- zoznámenie s knižnicou PCL
- načítanie mračien bodov
- vlastnosti mračna bodov
- vytvorenie databázy modelov
- spracovanie scény
- párovanie deskriptorov
- zhlukovanie korešpondencií
- odhad transformácie
- vizualizácia výsledkov

Software teda slúži ako kompletný rozpoznávací reťazec. Znamená to, že vykoná všetky kroky, ktoré v konečnom dôsledku povedú k rozpoznaniu objektu v scéne (pokiaľ sa tam objekt nachádza a bude možné ho rozpoznať). Pred spustením sa dajú nastaviť rôzne parametre príkazového riadu, ktoré dovoľujú užívateľovi meniť používané algoritmy (a ich parametre) pri rozpoznávaní, vrátane využitia experimentálnych zmien, ktorým je venovaná kapitola 5.

Ukážky výsledkov v jednotlivých krokoch spracovania mračien sú prezentované na dátovej sade, ktorá obsahuje model a scénu valca a pochádza zo snímacieho zariadenia Microsoft Kinect. Dátová sada mi bola poskytnutá vedúcim práce.



Obr. 10: Ukážka dátovej sady valca. Vľavo model, vpravo scéna obsahujúca model a ďalšie body. Na tejto dátovej sade sú prezentované výsledky jednotlivých krokov pri rozpoznávaní objektu v scéne.

## 4.1 Point Cloud Library

Point Cloud Library (skrátene **PCL**) [13] je plne šablónová, multiplatformová open-source knižnica, napísaná v jazyku C++. Slúži na spracovanie 2D/3D obrázkov a mračien bodov. Obsahuje niekoľko State of the Art algoritmov a nástrojov pre spracovanie 3D dát. Knižnica je vytvorená s dôrazom na rýchlosť. Definované dátové štruktúry využívajú SSE optimalizácie.

Niektoré algoritmy majú svoju paralélnu implementáciu pomocou OpenMP direktív a súčasťou knižnice je aj rozšírenie na architektúru CUDA. Väčšina matematických operácií je implementovaná pomocou knižnice Eigen [14], ktorá slúži pre výpočty lineárnej algebry. Základ pre rýchle vyhľadávanie na základe k-susedov poskytuje FLANN (Fast Library for Approximate Nearest Neighbors) [15].

Všetky moduly a algoritmy v PCL presúvajú dáta pomocou zdieľaných ukazovateľov v knižnici Boost. Knižnica je rozdelená do menších knižníc, ktoré je možné kompilovať samostatne. Toto rozdelenie je dôležité pri distribúcii do zariadení, ktoré disponujú s menšími výpočtovými a

pamäťovými možnosťami. Rozdelenie do modulov je nasledovné: filtrovanie, segmentácia, odhad deskriptorov, kľúčové body, registrácia, rozpoznanie, k-d strom, oktánový strom, rekonštrukcia povrchu, vstupno-výstupný modul, vizualizácia.

V prílohe A.1 je detailne popísaný postup inštalácie na operačnom systéme Ubuntu.

## 4.2 Načítanie mračien bodov

Po spustení programu je potrebné načítať mračná bodov do dátových štruktúr, aby s nimi bolo možné ďalej pracovať. Vďaka PCL je možné využiť vstupno-výstupný modul, ktorý poskytuje metódy na čítanie mračien bodov z rôznych druhov 3D skenerov a senzorov, alebo z nasledovných formátov:

- PCD
- PLY
- IFS

Všetky tri formáty je možné načítať pomocou metódy `pcl::io::load`, ktorá na základe prípony súboru (predaného ako prvý parameter funkcie) zavolá správnu metódu a načíta obsah dátového súboru do štruktúry `pcl::PointCloud<T>`, ktorá reprezentuje základnú triedu mračna bodov. Parameter `T` označuje konkrétny dátový typ bodu, ktorý má byť zo súboru načítaný. Je potrebné tento typ špecifikovať pri deklarácii. V prípade, že sa budeme snažiť napríklad načítať farebné body z dátovej sady, ktorá farebnú zložku bodov neobsahuje, metóda vyhodí chybovú hlášku. Medzi najpoužívanejšie typy bodov patria:

- `pcl::PointXYZ`
- `pcl::PointXYZRGB`
- `pcl::PointXYZRGBA`

V mojej aplikácii načítavam model scény a modelu, ktoré sú predané pomocou argumentov príkazového riadku. Používaný typ bodu je `pcl::PointXYZRGBA`. Táto dátová štruktúra obsahuje súradnice X, Y, Z a alfa kanál A nesúci informáciu o priehľadnosti.

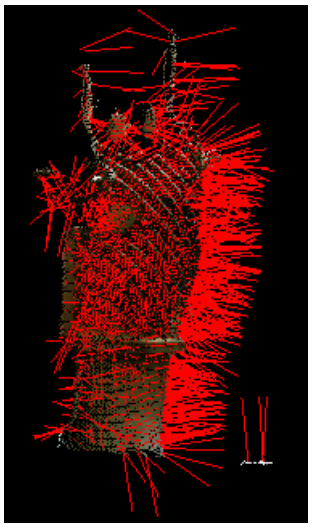
Tento typ bodu som zvolil preto, aby som mohol pracovať s farebnou zložkou snímaných mračien.

## 4.3 Vlastnosti mračna bodov

Každé mračno bodov, či už sa jedná o jeden z modelov, alebo o scénu, musí obsahovať tieto vypočítane vlastnosti:

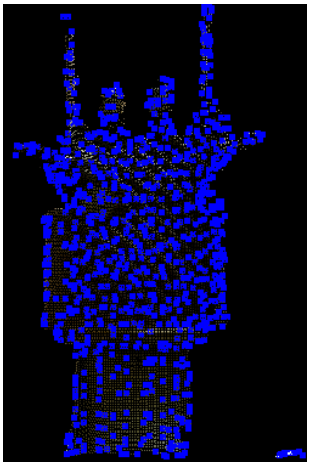
- povrchové normály
- kľúčové body
- deskriptory

Výpočet **povrchových normál** pre každý bod v mračne je zásadný krok, pokiaľ sa snažíme odhadnúť 3D deskriptory, ktoré sú na nich závislé. V knižnici PCL som využil triedu `pcl::NormalEstimationOMP`, ktorá vykonáva odhad rovnakým spôsobom, ako je detailne popísané v kapitole 3.2 . Vybraná trieda využíva paralelizáciu pomocou OpenMP direktív. Ako príklad autori uvádzajú, že ak je táto trieda využitá na počítači s 8-jadrovým procesorom, odhad normál by mal byť o 6 až 8 krát rýchlejší.



Obr. 11: Vizualizácia normál vypočítaných v modele valca. Pri odhade normál bolo použité vyhľadávanie 16 najbližších susedov.

Ďalším dôležitým krokom, ktorý výrazne ovplyvňuje rýchlosť výpočtu deskriptorov a celkového rozpoznávania, je získanie bodov záujmu, taktiež nazývaných ako **kľúčové body**. Z týchto bodov následne počítame deskriptory. K získaniu týchto bodov je možné použiť niektorý z dostupných algoritmov, alebo získať body podvzorkovaním mračna bodov. Ja som použil algoritmus podvzorkovania, ktorý je dostupný v knižnici PCL v triede `pcl::UniformSampling`. Detailnejší popis oboch spôsobov získania kľúčových bodov a taktiež popis algoritmu Uniform Sampling sa nachádza v kapitole 3.3.



Obr. 12: Vizualizácia kľúčových bodov získaných pomocou algoritmu Uniform Sampling. Veľkosť listu bola nastavená na 1 cm a počet výsledných bodov bol 874.

Posledným krokom je samotný **výpočet deskriptorov** pre každý kľúčový bod v danom mračne bodov. Pri vývoji som vďaka knižnici PCL mohol využiť rôzne State of the Art algoritmy, ktoré knižnica implementuje, a porovnať ich výsledky. Hlavné hodnotiace kritéria sú schopnosť algoritmu nájsť správnu výslednú transformáciu objektu a čas, za ktorý boli deskriptory odhadnuté pre dané mračno bodov. Prvé kritérium je skôr subjektívneho charakteru, pretože správnosť dosiahnutého výsledku bude daná tým, že nebudú pri pozorovaní viditeľné žiadne chyby vo výslednej transformácii objektu. Pri vývoji boli testované nasledujúce algoritmy:

- Signature of Histograms of OrienTations (SHOT)
- Point Feature Histogram (PFH)
- Fast Point Feature Histogram (FPFH)

Výsledky a porovnania algoritmov sú prezentované v kapitole 4.9.

## 4.4 Vytvorenie databázy modelov

Po spustení programu je potrebné vytvoriť databázu modelov, ktorá by mala obsahovať pre každý objekt vlastnosti popísané v kapitole 4.3.

V prípade, že sa používajú pre rozpoznávanie rovnaké parametre pre výpočet normál, kľúčových bodov a deskriptorov, je možné uložiť databázu modelov do súboru a pri spustení ju iba správne načítať.

Moja implementácia neobsahuje vyššie popísanú databázu modelov, pretože sa rozpoznáva jeden model v jednej scéne. Pri spustení programu sú načítané dve mračná bodov, tvoriace model a scénu. Vlastnosti modelu sú vypočítané pri spustení a následne sa ihneď hľadá objekt v načítanej scéne.

Výpočet vlastností pre model po spustení programu je vykonaný z toho dôvodu, aby boli výsledky uložené do pamäti a dostupné počas behu programu vždy pri zmene scény. Zmenu scény je v mojom programe možné vykonať pomocou stlačenia klávesy **r**. Scéna sa otočí o uhol $2\pi$. Následne sa opakuje proces rozpoznávania, avšak už sa nepočítajú vlastnosti pre model, ale iba pre scénu. Pomocou nich sa v zápätí hľadá výskyt modelu v rotovanej scéne. Správnym nájdením modelu v takto modifikovanej scéne je dokázaná odolnosť deskriptorov voči rotácii.

## 4.5 Spracovanie scény

V tomto kroku musíme spracovať mračno bodov obsahujúce snímanú scénu. Tá môže byť dodaná zo súboru, alebo zo snímacieho zariadenia.

Pre scénu sa musia vypočítať všetky vlastnosti popísané v kapitole 4.3, rovnakým spôsobom ako pre modely. V mojej aplikácii je scéna dodaná zo súboru pomocou parametru príkazového riadku. Za behu programu je možné rotovať mračno bodov o uhol $2\pi$ pomocou klávesy **r** - po rotácii je scéna znova spracovaná a začne proces hľadania modelu v rotovanej scéne.

Pomerne jednoduchou úpravou by bolo možné upraviť beh aplikácie tak, aby sa mračná bodov získavali v reálnom čase zo snímacieho zariadenia. Knižnica PCL poskytuje vo vstupno-výstupnom module funkcie na získavanie týchto dát priamo zo zariadení kompatibilných s frameworkom OpenNI, medzi ktoré patria napríklad zariadenia:

- Microsoft Kinect
- Asus Xtion Pro

Táto úprava nebola vykonaná z toho dôvodu, že som nemal k dispozícii potrebný hardware.

## 4.6 Párovanie deskriptorov

Po vypočítaní lokálnych deskriptorov pre scénu nasleduje hľadanie možných dvojíc korešpondenčných bodov, ktoré by mohli predstavovať správnu transformáciu modelu v scéne.

K ukladaniu dvojíc korešpondenčných bodov medzi modelom a scénou sa v knižnici PCL používa štruktúra `pcl::Correspondence`, ktorá sa skladá z troch vlastností:

- index bodu scény
- index bodu modelu
- vzdialenosť medzi bodmi

V tomto kroku je tvorený vektor korešpondencií, postupom detailnejšie popísaným v kapitole 3.5. Výsledný vektor je reprezentovaný dátovou štruktúrou `pcl::Correspondences`. Zoznam sa následne spracováva v ďalšom kroku.

## 4.7 Zhlukovanie korešpondencií

Knižnica PCL ponúka niekoľko tried na zhlukovanie korešpondencií. Pri vývoji som testoval dva algoritmy, konkrétne **Houghovo hlasovanie** a **Geometrická konzistencia**. Pozorovaním oboch algoritmov som prišiel na to, že výsledky algoritmu, ktorý zachováva geometrickú konzistenciu medzi nájdenými korešpondenciami, poskytuje presnejšie výsledky. V PCL je implementovaný v rámci triedy `pcl::GeometricConsistencyGrouping` a pri testovaní na rôznych dátových sadách s rôznym počtom nájdených korešpondencií sa čas vykonania pohyboval v rozmedzí 1-3ms. Vo všetkých prípadoch boli výsledky dostačujúce, preto sa v tejto práci nezaoberám vylepšením algoritmov patriacich do tohto kroku. Detailnejší popis zhlukovacích algoritmov sa nachádza v kapitole 3.6 .

## 4.8 Nastavenie parametrov algoritmov

Použitie správnych parametrov algoritmov je dôležitá súčasť rozpoznávacieho softwaru. Každý typ rozpoznávaných objektov potrebuje trochu iné parametre. Ak by sme sa napríklad zamerali na odhad povrchových normál, tak si musíme premyslieť, aký veľký nastavíme počet susedov $k$, ktorí sa budú podieľať na výpočte normály v danom bode (pri zvolení polomeru $r$ je situácia obdobná). Keď vyhľadávame objekt, kde potrebujeme zachytiť aj tie najmenšie detaily, naše $r$ alebo $k$ by mali byť malé, v opačnom prípade by sme mali použiť väčšie hodnoty.

Pri vývoji aplikácie som skúsil rôzne nastavenia parametrov, aby som zistil, ktoré hodnoty mi dávajú najlepšie výsledky. Hodnoty som zisťoval empiricky, pričom doporučené hodnoty parametrov som získal z komunity užívateľov používajúcich knižnicu PCL.

Výsledné odhady parametrov som rozdelil na dve časti. Dôvodom bolo zistenie, že algoritmy patriace do skupiny PFH (FPFH, PFHRGB) potrebujú iné parametre polomerov používaných pri hľadaní najbližších susedov v okolí bodov, pre ktoré počítame deskriptory. Detailnejšie vyhodnotenie rôznych deskriptorov sa nachádza v kapitole 4.9.

Nie sú uvedené nastavenia hodnôt pre získanie kľúčových bodov podvzorkovaním, pretože sa mi overilo použitie hodnôt 0.01 pre model a 0.03 pre scénu, výsledkom ktorých bol rozumný počet kľúčových bodov pre spracovanie. Taktiež neuvádzam nastavenie maximálnych povolených vzdialeností a počet susedov použitých pri párovaní deskriptorov, pretože kombinácia maximálnej vzdialenosti 50 a k = 1 sa ukázala ako najlepšia z testovaných. Nastavenie počtu $k$ susedných bodov zahrnutých do výpočtu normál som ponechal na hodnote $k = 16$. Posledné nastavenia, ktoré ďalej neuvádzam, sú nastavenia zhlukovacieho algoritmu. Konkrétne najmenší počet korešpondencií, ktorý je potreba pri zhlukovaní (minimum pre získanie 6 DoF pozície je 3), ja pracujem s hodnotou 5. Minimálna veľkosť zhluku bola nastavená na hodnotu 0.01.

Tabuľka 1 ukazuje rôzne konfigurácie rozpoznávania objektov pre prvú skupinu deskriptorov (PFH, FPFH a PFHRGB). Z uvedených konfigurácií som si vybral variantu číslo 3, ktorú som následne používal pri testovaní a experimentoch.

| | Deskriptory | Výsledky |
|---|---|---|
| 1 | r = 0.02 | Presnejší odhad, ale vysoký čas pre použitie v praxi |
| 2 | r = 0.005 | Vyšší čas odhadu, výsledky podobné ako varianta 3 |
| 3 | r = 0.009 | Prijateľná kombinácia presnosti a rýchlosti |

Tabuľka 1: Rôzne varianty nastavenia parametrov pri použití algoritmov zo skupiny, do ktorej patria deskriptory PFH, FPFH, PFHRGB

Pre druhú skupinu deskriptorov (SHOT, SHOTCOLOR) som vybral z tabuľky 2 variantu číslo 1. Základnú hodnotu polomeru použitú v tejto variante som zistil empiricky. Táto hodnota bola skoro hraničná, tzn. nastavením hodnôt nižších ako 0.018 nebol hľadaný objekt rozpoznaný.

| | Deskriptory | Výsledky |
|---|---|---|
| 1 | r = 0.019 | Prijateľná kombinácia presnosti a rýchlosti |
| 2 | r = 0.03 | Vysoký čas odhadu, vhodnejší pre oba algoritmy |
| 3 | r = 0.4 | Presné výsledky, vysoký čas odhadu |

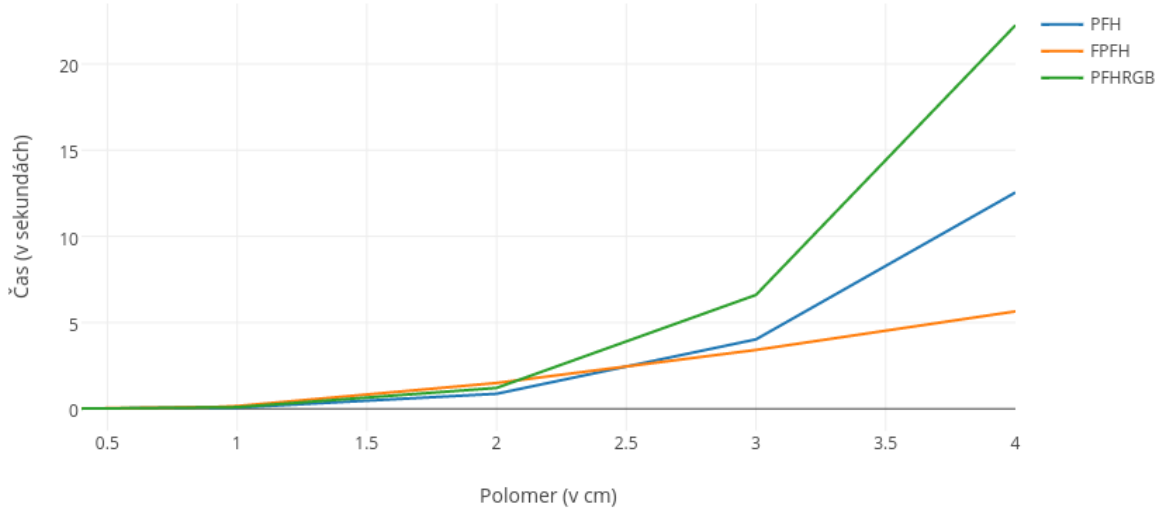
Tabuľka 2: Rôzne varianty nastavenia parametrov pri použití algoritmov zo skupiny, do ktorej patria deskriptory SHOT, SHOTCOLOR

## 4.9 Vyhodnotenie algoritmov

Testovanie prebiehalo na dátovej sade, ktorá obsahuje model a scénu valca. Výsledné časy v tejto kapitole sú získané pri odhade deskriptorov pre scénu valca. Toto mračno bodov obsahuje 29706 bodov a podvzorkovaním bolo získaných 827 kľúčových bodov, pre ktoré sa následne počítali deskriptory. Inými slovami, výsledné časy sa rovnajú času strávenému pri odhade 827 deskriptorov.

Získané výsledky sú rozdelené na dve časti, pretože algoritmy SHOT a SHOTCOLOR potrebovali iné parametre, ako algoritmy PFH, FPFH a PFHRGB (viď kapitola 4.8).

Na obrázku 13 je zobrazený rast časovej náročnosti algoritmov PFH, FPFH a PFHRGB so zvyšujúcim sa polomerom, ktorý je používaný pri výpočte deskriptorov (konkrétne pri hľadaní najbližších susedov v okolí bodu, viac v kapitole 3.4.1.1). Ako je z obrázku vidieť, rast deskriptorov PFH a PFHRGB je kvadratický, pričom deskriptor FPFH rastie lineárne.

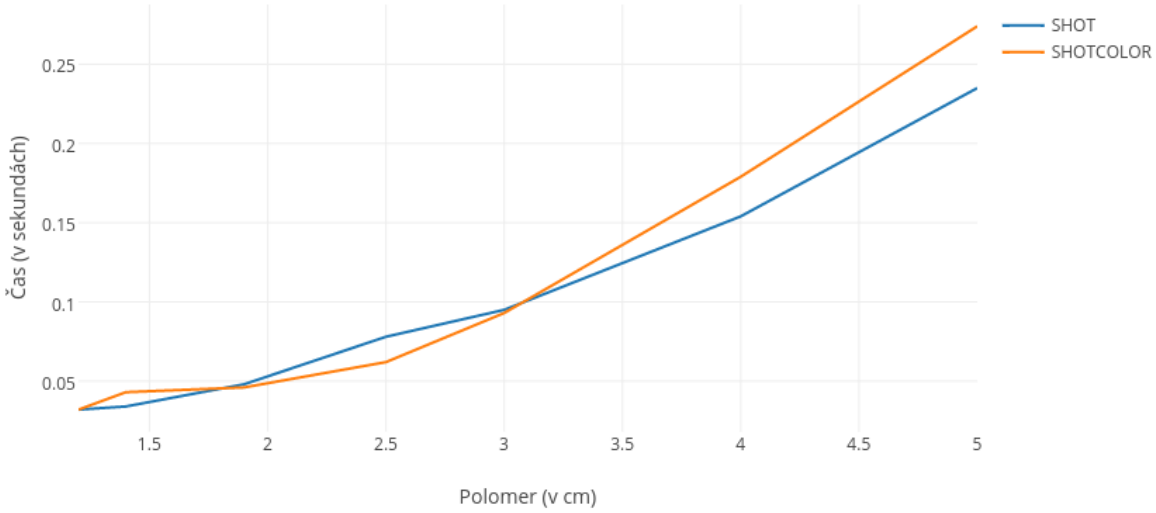



Obr. 13: Porovnanie časovej náročnosti algoritmov PFH, FPFH a PFHRGB

Rast časovej náročnosti algoritmov patriacich do druhej skupiny testovaných algoritmov, pozostávajúcej z deskriptorov SHOT a farebnej varianty SHOTCOLOR, je zobrazený na obrázku 14. Z grafu je možné zistiť, že rast oboch deskriptorov je lineárny. Bohužiaľ ich nevýhodou je vysoký počet dimenzií, preto by neboli vhodné pre moje ďalšie experimenty.

Ak teda zoberiem do úvahy relatívne nízky počet dimenzií deskriptorov FPFH (konkrétne 33) a lineárny rast časovej náročnosti algoritmu, rozhodol som sa pri experimentovaní používať tento druh deskriptorov. V experimentoch týkajúcich sa redukcie dimenzií pomocou analýzy hlavných komponentov (viď kapitola 5.1) budem používať aj algoritmy PFH a PFHGRB. Dô-

vodom je v tomto prípade nižší počet dimenzií ako obsahovala druhá skupina algoritmov SHOT a SHOTCOLOR.



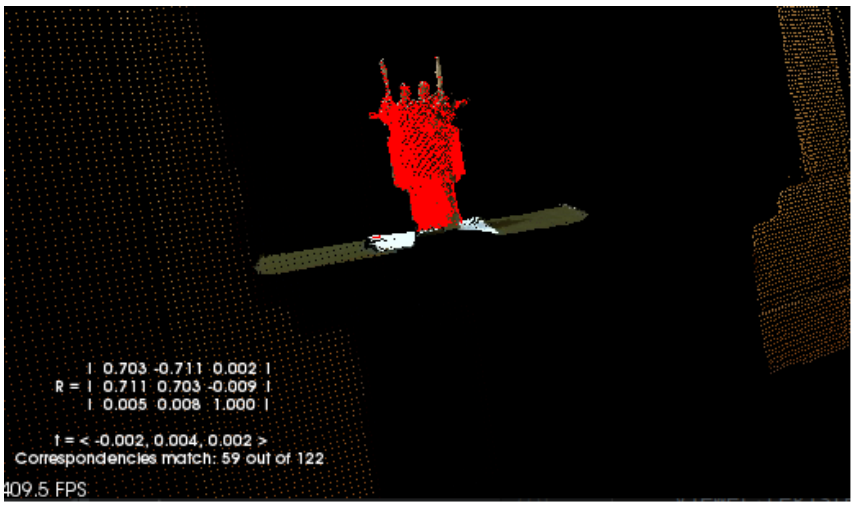

Obr. 14: Porovnanie časovej náročnosti algoritmov SHOT a SHOTCOLOR

## 4.10 Vizualizácia výsledkov

Vďaka vizualizačnému modulu v knižnici PCL je možné jednoducho vykresľovať výsledky algoritmických operácií nad 3D mračnami bodov. Implementuje rozsiahlu vrstvu pre potreby N-dimenzionálnej vizualizácie nad existujúcim open-source riešením pre vykresľovanie 3D mračien bodov - **Visualization Toolkit (VTK)** [16]. Medzi základné funkcie vizualizačného modulu v knižnici PCL patria:

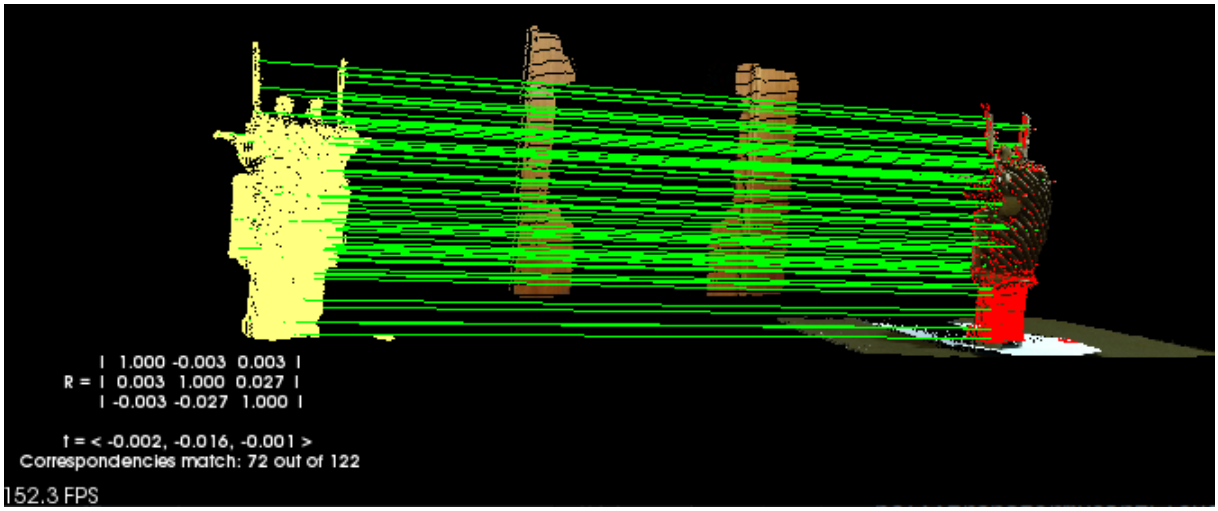
- metódy pre vykresľovanie a nastavovanie vizuálnych vlastností (farby, veľkosti bodov, viditeľnosť a pod.) pre rôzne N-dimenzionálne dátové sady, ktoré sú vo formáte mračna bodov pcl::PointCloud<T>
- metódy pre kreslenie základných 3D tvarov (napr. válce, gule, čiary, polygóny a pod.) z množín bodov, alebo z rôznych parametrických rovníc
- možnosti vizualizácie histogramu (modul PCLHistogramVisualizer)
- zobrazovanie geometrie (napr. vypočítané normály) z 3D mračna bodov

Po rozpoznaní je vykreslená scéna, ktorá obsahuje červenou farbou vyznačený nájdený objekt. V ľavom dolnom rohu je vypísaná rotačná matica a vektor posunu. Na obrázku 15 je názorný príklad pri rozpoznávaní testovacej scény valca.




Obr. 15: Rozpoznaný model valca v snímanej scéne

Pri spustení mojej aplikácie je možné zobraziť okrem nájdeného objektu taktiež korešpondencie, ktoré sa medzi modelom a scénou pri hľadaní našli (viď príloha A.2). Príklad vizualizácie nájdených korešpondencií je na obrázku 16.




Obr. 16: Rozpoznaný model valca v snímanej scéne s dodatočne vykreslenými korešpondenciami

# 5 Experimenty

Táto kapitola je venovaná mojim navrhnutým vylepšeniam a experimentálnym overením ich funkčnosti. Všetky vylepšenia sú overené na dátových sadách, ktoré obsahujú tieto snímané predmety:

- valec
- krhla
- krabica
- zubná pasta
- mlieko

Prvým vylepšením je redukcia dimenzií deskriptorov (konkrétne PFH, FPFH a PFHRGB) pomocou analýzy hlavných komponentov (PCA), pretože ich implementácia v knižnici PCL odhaduje deskriptory s vysokým počtom hodnôt (viď kapitola 5.1). Očakávaným výsledkom je zjednodušenie procesu párovania deskriptorov, čo by mohlo viesť k zrýchleniu celého procesu rozpoznávania objektu. V kapitole 5.1.1 popisujem, akým spôsobom som získal vhodné počty redukovaných dimenzií pre vyššie uvedené deskriptory. Postup, akým som do procesu rozpoznávania objektov aplikoval analýzu hlavných komponentov a následné detaily implementácie popisujem v kapitolách 5.1.2 a 5.1.3. Dosiahnuté výsledky experimentov vykonaných na testovacích dátových sadách sú detailne popísané v kapitole 5.1.4.

Druhou navrhnutou zmenou je modifikácia odhadu deskriptorov FPFH v knižnici PCL tak, aby tvoril histogramy, ktoré majú už pri výpočte menší počet dimenzií, čo má za následok vytvorenie nízko dimenzionálnych deskriptorov. Tejto modifikácii je venovaná kapitola 5.2.

Pri objavení deskriptorov, ktoré kódujú okrem geometrie aj farebnú zložku (napr. PFHRGB) ma napadlo, že by mohol byť odhad FPFH deskriptorov so zakomponovaním farebnej zložky RGB presnejší vo farebných scénach, ako pôvodný deskriptor. Preto mojou ďalšou navrhnutou a experimentálne overenou zmenou je rozšírenie algoritmu FPFH tak, aby kódoval farebnú zložku RGB (pracovne ho nazývam FPFHCOLOR). Detailnému postupu je venovaná kapitola 5.3 a výsledky sú dostupné v kapitole 5.3.1. Následne popisujem v kapitole 5.3.2 použitie PCA analýzy na takto modifikovaný deskriptor aby som zistil, či farebná zložka RGB hrá rolu pri popise okolia bodu.

Poslednou navrhnutou zmenou je vytvorenie algoritmu, ktorý hľadá najbližších susedov pomocou hrubej sily. Motiváciou bola prípadná budúca paralelizácia na GPU, aby som mohol porovnať efektívnosť takto paralelizovaného algoritmu a k-d stromu, ktorý sa bežne využíva na tieto účely. Tejto modifikácii je venovaná kapitola 5.4, ktorá obsahuje porovnanie času behov týchto algoritmov pri sériovom spracovaní.

## 5.1 Redukcia dimenzií deskriptorov

V praxi sú 3D deskriptory väčšinou reprezentované dátovými štruktúrami, ktoré sú zložené z veľa čísel. Každé číslo reprezentuje jednu dimenziu deskriptoru. Ak si zoberieme napríklad implementáciu týchto algoritmov v knižnici PCL, tak zistíme, že väčšina z nich odhaduje vysoko dimenzionálne deskriptory (viď tabuľka 3).

Efektivita vyhľadávania $k$ najbližších susedov sa so zvyšujúcou dimenziou dát znižuje a blíži sa ku klasickému lineárnemu prehľadávaniu s časovou zložitosťou $O(n)$. Tento problém sa dá riešiť dvomi spôsobmi.

Prvý spôsob zahrňuje využitie aproximačných metód, ktoré v rozumnom čase poskytnú dostatočne dobrý odhad najbližšieho suseda. Väčšinou tieto metódy vrátia konkrétneho najbližšieho suseda (približne v 95% prípadoch) [15], ale tento poznatok je ovplyvnený dátovou sadou, v ktorej vyhľadávame. PCL používa knižnicu Fast Library for Approximate Nearest (FLANN), ktorá využíva tieto aproximačné metódy:

- k-d strom využívajúci náhodu
- hierarchický k-d strom

Každá z vyššie uvedených aproximačných metód je efektívnejšia na iných dátových sadách, preto FLANN obsahuje rozhodovací algoritmus, ktorý na základe zloženia dát zvolí správnu aproximačnú metódu.

Druhou možnosťou je redukcia dimenzií dát takým spôsobom, aby sme zachovali čo najviac pôvodnej informácii. Najznámejšia metóda slúžiaca k redukcii dimenzií je **analýza hlavných komponentov (PCA)**. Detailnému popisu algoritmu je venovaná kapitola 3.8. V mojej aplikácii využívam túto analýzu k dvom účelom:

1. výpočet dimenzií redukovaných deskriptorov so zachovaním určeného percenta pôvodných informácií
2. redukcia dimenzií rôznych lokálnych deskriptorov s následným vyhodnotením výsledkov

| Názov algoritmu | Počet dimenzií |
|---|---|
| Point Feature Histogram (PFH) | 125 |
| Point Feature Histogram s farebnou zložkou RGB (PFHRGB) | 250 |
| Fast Point Feature Histogram (FPFH) | 33 |
| Signatures of Histograms Orientations (SHOT) | 9+352 |
| Signatures of Histograms Orientations s farebnou zložkou (SHOTCOLOR) | 9+1344 |

Tabuľka 3: Počet dimenzií rôznych lokálnych deskriptorov v knižnici PCL

### 5.1.1 Výpočet dimenzií redukovaných deskriptorov

V prípade, že chceme redukovať počet dimenzií odhadnutých deskriptorov potrebujeme vedieť, koľko dimenzií môže byť zredukovaných pri zachovaní určitého percenta informácií obsiahnutých v pôvodných deskriptoroch. Túto informáciu môžeme získať tak, že odhadneme lokálne deskriptory mračna bodov, ktoré reprezentuje model objektu. Následne ich transformujeme do matice rozmerov $n \times p$, kde $n$ je počet odhadnutých deskriptorov (resp. počet kľúčových bodov, pre ktoré boli deskriptory vypočítané) a $p$ je počet dimenzií pôvodných deskriptorov. Následne je vykonaná PCA analýza a vypočítaný vektor súhrnných energií $g$, pomocou ktorého vypočítame redukovaný počet dimenzií, pri zachovaní 95% pôvodných informácií (viď kapitola 3.8.7).

Pri mojich experimentoch som vypočítal množstvo redukovaných dimenzií (so zachovaním 95% pôvodných informácií) pre algoritmy PFH, FPFH, PFHRGB na piatich rôznych dátových sadách. V tabuľke 4 sú výsledky zahrňujúce informácie o pôvodných počtoch dimenziách jednotlivých deskriptorov, nasledované vypočítanými novými hodnotami dimenzií pre každú dátovú sadu. Ďalej uvádzam mediány hodnôt testovaných deskriptorov, ktoré budú následne v kapitole 5.1.2 použité pre zhodnotenie presnosti a porovnania časov pred a po redukcii s použitím hodnôt získaných v tomto kroku.

| Názov algoritmu | Počet pôvodných dimenzií | Počet redukovaných dimenzií v dátových sadách [a] | Medián počtu dimenzií | Čas behu PCA analýzy |
|---|---|---|---|---|
| FPFH | 33 | 10, 9, 5, 10, 9 | 9 | $< 1$ ms [b] |
| PFH | 125 | 12, 9, 3, 8, 7 | 8 | 22 ms |
| PFHRGB | 250 | 16, 13, 9, 13, 12 | 13 | 40 ms |

[a] V nasledujúcom poradí: valec, krhla, krabica, zubná pasta, mlieko
[b] nameraná hodnota je pravdepodobne pod rozlišovaciu schopnosť môjho merania

Tabuľka 4: Počty dimenzií rôznych deskriptorov po redukcii pomocou PCA analýzy

Ako je na prvý pohľad vidieť v tabuľke 4, pomocou PCA analýzy bolo možné zredukovať pôvodné počty dimenzií vypočítaných deskriptorov a to nasledovne: FPFH (72.72%), PFH (93.6%), PFHRGB (94.8%).

Výrazná redukcia dimenzií bola možná zo samotnej podstaty odhadu deskriptorov. Po hlbšej analýze odhadovaných deskriptorov som zistil, že pomerne veľká časť z čísel tvoriacich daný deskriptor, sú hodnoty 0, ktoré sa žiadnym spôsobom nepodieľajú na výsledku ďalšieho spracovania. Z toho vyplýva, že vstupná matica použitá ako vstup PCA analýzy je riedka, tzn. má väčšinu prvkov nulových.

Ďalší zaujímavý poznatok sú samotné hodnoty redukovaných deskriptorov. Konkrétne plávajúca hodnota dimenzií v rôznych dátových sadách pri použití rovnakého deskriptoru. Táto

variabilita nastáva z dôvodu zložitosti scény - čím zložitejšia je scéna, tým viac dimenzií by sme mali potrebovať po redukcii.

Taktiež vidíme, že medián počtu dimenzií je najvyšší pri použití deskriptoru PFHRGB, ktorý kóduje okrem geometrie taktiež farebnú zložku RGB. Avšak nárast počtu dimenzií nie je priamo úmerný pôvodnej dimenzii deskriptoru, pretože bolo stále možné zredukovať väčšinu dimenzií, takže v porovnaní s variantou bez farby (PFH) bol nárast iba o 4 dimenzie.

Všetky mediány počtov dimenzií v tabuľke 4 budú v nasledujúcej kapitole použité pre redukciu deskriptorov a bude overená presnosť výsledku rozpoznávania pri použitých takto zredukovaných deskriptorov a porovnané časy výpočtu.

### 5.1.2 Použitie redukovaných deskriptorov pri rozpoznávaní

Aby bolo možné overiť správnosť a použiteľnosť výpočtov vykonaných v kapitole 5.1.1, je potrebné ich použiť pri redukcii jednotlivých typov deskriptorov. Tieto redukované deskriptory budú po redukcii použité pri párovaní deskriptorov (viď kapitola 3.5).

Ak si zoberieme postupnosť krokov, ktoré sa musia vykonať k rozpoznaniu objektu pri použití lokálnych deskriptorov (viď kapitola 3.4.1), tak je nutné po výpočte deskriptorov v kľúčových bodoch pridať ďalší krok, v ktorom deskriptory redukujeme.

Pred samotnou redukciou som sa musel zamyslieť, akým spôsobom sa bude PCA analýza používať. Pri prvých pokusoch som si neuvedomil, že ak budem aplikovať analýzu zvlášť pre model a pre scénu (tj. výpočet bázových vektorov z vlastných vektorov pre každý typ mračna bodov zvlášť), prepočítaval by som deskriptory do iných súradnicových sústav a následné použitie by vracalo nepoužiteľné výsledky, resp. žiadna použiteľná transformácia by sa nenašla. Po týchto neúspechoch som postup zmenil nasledovne:

1. Po spustení programu sa **spracuje model** spôsobom popísaným v kapitole 4.4. Z deskriptorov patriacich modelu sa následne vytvoria nové deskriptory s menším počtom dimenzií (nový počet dimenzií pre daný typ deskriptoru je daný výsledkami v tabuľke 4)

2. Po vykonaní kroku, v ktorom sa počítajú vlastnosti scény (viď kapitola 4.5) sa následne vypočítajú redukované deskriptory pre **scénu** (pomocou bázových vektorov získaných pri redukcii modelu v kroku 1)

3. Následne sa redukované deskriptory modelu a scény použijú pri párovaní deskriptorov

Je dôležité poznamenať, že vďaka vyššie popísaným krokom sa PCA analýza aplikuje iba na modely a to iba raz, pri spustení programu, resp. vytváraní databázy modelov. Následné redukovanie deskriptorov z scény (vykonané vždy pri zmene scény buď funkciou otočenia v mojej aplikácii, alebo dátami zo snímacieho zariadenia) zahrnuje iba násobenie matice bázových vektorov s maticou obsahujúcou pôvodné deskriptory. Matica bázových vektorov je vytvorená z $L$ vlastných vektorov (kde $L$ je nová redukovaná dimenzia), ktorým zodpovedajú najvyššie vlastné čísla. Vlastné čísla a vektory boli získané z kovariančnej matice modelu (viac v kapitole 3.8).

Výsledky rozpoznania objektu v scéne pomocou redukovaných deskriptorov sú popísané v kapitole 5.1.4.

### 5.1.3 Implementácia algoritmu PCA

Implementácia algoritmu prebiehala v programovacom jazyku C++. Vytvoril som šablónu triedy, ktorú som nazval `GenericPCA`. Aktuálne šablóna pracuje so všetkými štruktúrami, ktoré sú vytvorené pomocou šablóny `pcl::Histogram<N>`. Táto šablóna predstavuje histogramy s variabilnou veľkosťou (napr. deskriptory PFH, FPFH a pod.). Jednoduchou úpravou, ktorá je zakomentovaná v kóde, je možné dosiahnuť generickosti a pracovať tak so všetkými dátovými štruktúrami v knižnici PCL (nie len histogramami) spôsobom, ktorý je vysvetlený v ďalšom odseku tejto kapitoly. Dôvod, prečo nie je použitá úplná generickosť algoritmu je ten, že autori knižnice PCL do tejto chvíle neopravili získavanie dát z tohto dátového typu histogramu, tým pádom by pri použití neboli správne získané všetky hodnoty histogramu. Túto chybu plánujú opraviť vo verzii 2.0 (aktuálna verzia je 1.8).

Pri definovaní nového dátového typu v knižnici PCL je nutné dať algoritmom vedieť, akým spôsobom môžu získavať dáta z novej štruktúry. Toto je dosiahnuté vytvorením triedy zo šablóny `DefaultPointRepresentation<T>`, ktorá sa nachádza v mennom priestore `pcl`. Parameter $T$ reprezentuje dátovú štruktúru, pre ktorú chovanie definujeme. V takto vytvorenej triede musíme definovať konštruktor, v ktorom nastavíme triednu premennú `nr_dimensions_` na požadovaný počet dimenzií, ktoré deskriptor obsahuje. Následne ešte definujeme metódu `copyToFloatArray`, ktorá zabezpečuje transformáciu deskriptoru do poľa typu `float`.

V prípade, že by sa v knižnici neobjavila chyba popísaná na začiatku tejto kapitoly, by som vďaka tomuto postupu bol schopný implementovať moju šablónu triedy tak, aby získavala dáta z rôznych dátových štruktúr definovaných v rámci knižnice PCL. V metódach, ktoré vyžadovali extrakciu dát z dátových štruktúr reprezentujúcich jednotlivé deskriptory by som inicializoval triedu `DefaultPointRepresentation<T>` s konkrétnym dátovým typom a následne by bolo možné použiť funkciu `copyToFloatArray`, ktorá ako prvý parameter prijíma deskriptor v pôvodnej dátovej štruktúre a druhý parameter obsahuje ukazovateľ na pole typu `float`, do ktorého by boli jednotlivé hodnoty deskriptora transformované. Moja šablóna triedy obsahuje dva parametre:

- PointIn
- PointOut

kde prvý parameter reprezentuje dátovú štruktúru, v ktorej sú uložené vypočítané deskriptory. Druhým parametrom určujeme dátový typ, do ktorého budú uložené deskriptory s redukovaným počtom dimenzií. Výstupná dátová štruktúra PointOut môže byť konkrétnou triedou šablóny `pcl::Histogram<N>`, kde parameter $N$ určuje dimenziu výsledného histogramu.

Šablóna triedy `GenericPCA` obsahuje nasledovné metódy (väčšinou volané v rovnakom poradí):

- `setInputModel` - transformuje vypočítané deskriptory patriace modelu do matice. Deskriptory musia byť uložené v šablóne triedy `pcl::PointCloud<T>`, kde $T$ je parameter určujúci konkrétny typ deskriptorov. Výsledná transformovaná matica má rozmery $n \times p$, kde $n$ je počet deskriptorov a $p$ je pôvodný počet dimenzií deskriptoru. Následne je použitím tejto matice vykonaná PCA analýza pre model a sú získané bázové vektory, ktoré sú uložené v matici $p \times p$

- `setInputScene` - táto metóda vykonáva transformáciu deskriptorov patriacich spracovávanej scéne zo šablóny triedy `pcl::PointCloud<T>` do matice rovnakým spôsobom, ako metóda `setInputModel`

- `setReducedDimensions` - z vlastných vektorov vypočítaných pre model vytvorí nové bázové vektory takým spôsobom, že sa vyberie $N$ vlastných vektorov s najvyšším vlastným číslom, kde parameter $N$ je predaný pri volaní funkcie

- `compute` - vykoná výpočet PCA analýzy pre scénu s použitím bázových vektorov vytvorených v metóde `setReducedDimensions`

- `getSuggestedDimensions` - metóda vypočíta redukovaný počet dimenzií pri zachovaní určitého percenta pôvodných informácií. Percento definujeme prahovou hodnotou, ktorá je predaná parametrom funkcie a náleží do intervalu $[0.0, 1.0]$ Výpočet je detailne popísaný v kapitole 3.8.7

- `getModelPoints` - metóda vráti redukované deskriptory modelu, ktoré sú uložené v konkrétnej triedy vytvorenej zo šablóny `pcl::PointCloud<T>`, kde paramter $T$ bude dátový typ vytvorený zo šablóny `pcl::Histogram<N>`, v ktorej parameter $N$ predstavuje počet redukovaných dimenzií deskriptorov

- `getScenePoints` - metóda vráti redukované deskriptory scény, rovnako ako predchádzajúca metóda `getModelPoints`

Konkrétne príklady použitia mojej šablóny triedy sú k dispozícii v prílohe A.3. Návod na vytvorenie vlastného dátového typu v knižnici PCL sa nachádza v ukážke 2.

### 5.1.4 Výsledky rozpoznávania po redukcii

Na základe poznatkov a výsledného postupu nachádzajúceho sa v kapitole 5.1.2 som do môjho programu, ktorý slúži pre rozpoznanie objektu na základe jeho modelu v RGB-D obraze implementoval krok, v ktorom sa redukujú pôvodné deskriptory na určitý počet dimenzií (viď tabuľka 4) a následne sa používajú pri ďalšom spracovaní. Výsledky sú prezentované v tabuľke 5. Z výsledkov je patrné, že čas strávený pri párovaní deskriptorov je niekoľkonásobne nižší, ako pred redukciou. Aj keď zoberieme do úvahy čas strávený redukciou (konkrétne sa jedná iba o násobenie matice bázových vektorov s maticou obsahujúcou pôvodné deskriptory zo scény), ktorý sa pohyboval v okolí 3 ms, stále sa použitie redukcie vyplatí, pretože ďalšie spracovanie je niekoľkonásobne rýchlejšie. Okrem toho sa výsledná transformácia objektu našla v 100% testoch, ktoré som vykonal pomocou redukovaných deskriptorov, takže nedošlo k znehodnoteniu pôvodných dát.

V mojom programe je možné použiť redukované deskriptory pri rozpoznávaní objektu v scéne. Predaním správnych parametrov je možné využiť PCA analýzu k redukovaniu nasledujúcich algoritmov: PFH, FPFH, PFHRGB a FPFHCOLOR (moja modifikćia pôvodného FPFH algoritmu, ktorému je venovaná kapitola 5.3). Detailnejší popis použitia správnych parametrov je dostupný v prílohe A.2.

| Názov deskriptoru (počet redukovaných dimenzií) | Dátová sada | Párovanie deskriptorov bez redukcie (v ms) | Párovanie deskriptorov po redukcii (v ms) |
|---|---|---|---|
| PFH (8) | Valec | 42 | 4 |
| | Krhla | 39 | 4 |
| | Krabica | 34 | 5 |
| | Zubná pasta | 10 | 1 |
| | Mlieko | 9 | 1 |
| FPFH (9) | Valec | 13 | 4 |
| | Krhla | 13 | 5 |
| | Krabica | 23 | 8 |
| | Zubná pasta | 6 | 1 |
| | Mlieko | 8 | 3 |
| PFHRGB (13) | Valec | 78 | 2 |
| | Krhla | 120 | 2 |
| | Krabica | 152 | 4 |
| | Zubná pasta | 31 | 1 |
| | Mlieko | 40 | 2 |

Tabuľka 5: Výsledky rozpoznávania pri použití deskriptorov redukovaných pomocou analýzy hlavných komponentov

## 5.2 Nízko dimenzionálny FPFH deskriptor

V knižnici PCL je FPFH deskriptor reprezentovaný 33 hodnotami. FPFH kóduje tri uhlové vlastnosti (vzdialenosť je zanedbaná), takže pre každú vlasnosť je vyhradených 11 hodnôt.

Nikde v literatúre som nenašiel dôvod, prečo si autori zvolili práve tieto hodnoty, preto som implementáciu algoritmu postupne upravoval tak, aby sa v každej modifikácii používal pre každú kódovanú vlastnosť nasledujúci počet hodnôt: 9, 7, 6, 4.

Rozpoznávanie objektov s použitím takto modifikovaného algoritmu FPFH som vykonal pre všetky testovacie scény (parametre algoritmov boli nastavené na hodnoty vybrané v kapitole 4.8). Výsledky dosiahnuté pri použití pôvodných a modifikovaných deskriptorov sú uvedené v tabuľke 6.

| Počet dimenzií deskriptoru (hodnôt pre každú vlastnosť) | Dátová sada | Počet nájdených korešpondencií | Počet korešpondencií vo výslednej transformácii |
|---|---|---|---|
| 33 (11) | Valec | 122 | 72 |
| | Krhla | 122 | 54 |
| | Krabica | 163 | 36 |
| | Zubná pasta | 51 | 10 |
| | Mlieko | 66 | 13 |
| 27 (9) | Valec | 125 | 73 |
| | Krhla | 128 | 50 |
| | Krabica | 172 | 33 |
| | Zubná pasta | 47 | 14 |
| | Mlieko | 71 | 16 |
| 21 (7) | Valec | 132 | 40 |
| | Krhla | 136 | 54 |
| | Krabica | 188 | 36 |
| | Zubná pasta | 65 | 12 |
| | Mlieko | 76 | 12 |
| 18 (6) | Valec | 157 | 63 |
| | Krhla | 178 | 38 |
| | Krabica | 233 | 8 |
| | Zubná pasta | 88 | 9 |
| | Mlieko | 116 | 8 |
| 15 (5) | Valec | 138 | 69 |
| | Krhla | 180 | 26 |
| | Krabica | 214 | 13 |
| | Zubná pasta | 88 | 6 |
| | Mlieko | 133 | 17 |

Tabuľka 6: Výsledky rozpoznávania pri použití pôvodných a nízko dimenzionálnych FPFH deskriptorov

Pri testovaní jednotlivých modifikácií algoritmu FPFH som zistil, že rozpoznanie objektu prebehlo korektne vo všetkých prípadoch. Výsledky sa líšili iba v počte korešpondencií patriacich do správnej transformácie. Objekt bol rozpoznaný aj pri použití 3 hodnôt pre každú vlastnosť (tj. 9 hodnôt histogramu z pôvodných 33), ktorú deskriptor kóduje. Tento prípad nie je zahrnutý vo výsledkoch, pretože sa vo väčšine prípadoch našla viac ako jedna inštancia modelu a do každej inštancie patrilo približne rovnaký počet korešpondencií (nebolo jednoznačne možné rozhodnúť, ktorá transformácia je správna).

Výsledky ukazujú, že počty korešpondencií patriacich do výslednej transformácie (ktorá predstavuje 6 DoF pozíciu objektu) sú pri použití pôvodných FPFH deskriptorov s **33 hodnotami** veľmi podobné ako pri použití modifikovanej verzie s **27 hodnotami**. Ako prijateľné sa ukazujú taktiež výsledky získané použitím **21 hodnôt**.

## 5.3 Rozšírenie algoritmu FPFH o farebnú zložku RGB (FPFHCOLOR)

Jedným z možných vylepšení algoritmu FPFH, ktorému je venovaná kapitola 3.4.1.2, je pridanie farebnej zložky, ktorá by mohla pomôcť k presnejšiemu popisu okolia bodu vo farebných mračnách bodov.

Je dôležité dodať, že knižnica PCL obsahuje deskriptor, ktorý sa nazýva PFHRGB. Tento deskriptor je rozšírením deskriptoru PFH takým spôsobom, aby deskriptor zachytával aj farebnú zložku RGB v okolí kľúčového bodu. K tomu je vyhradených 125 hodnôt histogramu, takže výsledný PFHRGB deskriptor tvorí 250 hodnôt, resp. dimenzií.

Táto vysoká dimenzia deskriptorov je nevýhodou a preto som sa rozhodol upraviť algoritmus FPFH (ktorý ma pôvodne 33 dimenzií) tak, aby zachytával farebnú zložku RGB v 33 hodnotách (tzn. pre každú farebnú zložku je vyhradených 11 hodnôt). Toto číslo som zvolil preto, že autori pôvodného FPFH deskriptoru použili 11 hodnôt pre každú zo zakódovaných uhlových vlastností tvoriacich deskriptor. Výsledný deskriptor, ktorý budem v zbytku práce nazývať **FPFHCOLOR**, sa teda skladá zo 66 hodnôt.

V mojej modifikácii sú zachované výpočty pôvodného FPFH deskriptoru, ku ktorým som pridal výpočet dodatočných troch hodnôt pre každú farebnú zložku R, G, B. Výpočet som prevzal z implementácie algoritmu PFHRGB implementovaného v knižnici PCL. Pre použitie som sa rozhodol z toho dôvodu, že pri testovaní PFHRGB na farebných scénach deskriptor fungoval celkom dobre a chcel som vidieť, ako bude výpočet fungovať v kombinácii s nízko dimenzionálnym deskriptorom FPFH. Výsledné hodnoty sú v poslednom kroku zlúčené do histogramu.

Výpočet nových vlastností deskriptoru pre každú farebnú zložku (vypočítavané vždy pre konkrétnu dvojicu bodov, viz kapitola 3.4.1.2) prebieha nasledovne:

1. pri výpočte sa používa dvojica bodov $P_1$ a $P_2$ a hodnota pre farebnú zložku R, ktorú som nazval $f_R$, je daná podmieneným vzorcom:

$$f_R = \begin{cases} \frac{P_{1R}}{P_{2R}}, & \text{ak } P_{2R} > 0.0 \\ 1.0, & \text{inak} \end{cases}$$

   kde $P_{1R}$ a $P_{2R}$ sú hodnoty farebnej zložky R pre bod $P_1$, resp. $P_2$.

2. následne je potrebné zabezpečiť, aby vypočítané hodnoty boli v intervale $[-1.0, 1.0]$, takže sa vykoná nasledujúci výpočet (v tomto prípade pre farebnú zložku R):

$$f_{R_{Final}} = \begin{cases} \frac{-1.0}{f_R}, & \text{ak } f_R > 1.0 \\ f_R, & \text{inak} \end{cases}$$

Pri experimentovaní budú vyhodnotené výsledky rozpoznávania v testovacích scénach použitím algoritmov FPFHCOLOR a FPFH. Následne bude algoritmus FPFHCOLOR redukovaný PCA analýzou, vďaka ktorej zistím, koľko redukovaných dimenzií je potrebných pri zachovaní 95% pôvodných informácií. Tento údaj mi poskytne informáciu o tom, či farebná zložka RGB zakódovaná do deskriptoru má nejaký význam pri popise okolia bodu, alebo nie.

### 5.3.1 Výsledky deskriptoru FPFHCOLOR

Výsledky rozpoznávani objekatu použitím pôvodného algoritmu FPFH a modifikovanej verzii, ktorá obsahuje farebnú zložku RGB, sú prezentované v tabuľke 7 .

Je vidieť, že algoritmus FPFH bol na testovaných dátach približne o 50% rýchlejší ako mnou modifikovaná verzia FPFHCOLOR. Toto číslo dáva podľa môjho názoru zmysel, pretože pri odhade FPFHCOLOR je vytvorených 2-krát viac histogramov, ako v pôvodnom algoritme FPFH.

Zaujímavý výsledok je vidieť, ak porovnáme nájdené korešpondencie (a patriace do výslednej transformácii) oboch algoritmov. Algoritmom FPFHCOLOR sa našiel lepší pomer nájdených a správne zaradených korešpondencií.

Vo farebnejších scénach (napr. zubná pasta a mlieko) sa pri použití FPFH našlo iba 10 a 12 korešpondencií patriacich do výslednej transformácie objektu. Pre porovnanie, pri použití farebnej varianty sa našlo 27 a 39 správnych korešpondencií (pri zachovaní rovnakých parametrov), pričom počet celkových nájdených korešpondencií bol podobný ako pri použití pôvodných deskriptorov. Farebná varianta teda potvrdila správnu transformáciu 2 až 3 násobným počtom korešpondencií.

V mojich experimentoch sa ukázalo, že deskriptory FPFHCOLOR sú presnejšie ako ich pôvodná verzia. Časová náročnosť bola získaná pri sériovom spracovaní, takže je možné zrýchliť čas odhadu týchto deskriptorov prípadnou paralelizáciou.

Získané výsledky sú samozrejme závislé na použitých dátových sadách, takže by bolo nutné experimenty zopakovať v rôznych scénach s variabilným osvetlením, ktoré by mohlo výsledky zhoršiť.

| Názov algoritmu | Dátová sada | Počet nájdených korešpondencií | Počet korešpondencií vo výslednej transformácii | Čas odhadu deskriptorov |
|---|---|---|---|---|
| FPFH | Valec | 122 | 72 | 88 ms |
| | Krhla | 120 | 54 | 70 ms |
| | Krabica | 163 | 36 | 84 ms |
| | Zubná pasta | 51 | 10 | 50 ms |
| | Mlieko | 65 | 12 | 27 ms |
| FPFHCOLOR | Valec | 108 | 87 | 167 ms |
| | Krhla | 130 | 85 | 89 ms |
| | Krabica | 190 | 153 | 160 ms |
| | Zubná pasta | 39 | 27 | 48 ms |
| | Mlieko | 61 | 39 | 57 ms |

Tabuľka 7: Výsledky rozpoznávania pri použití deskriptorov FPFH a FPFHCOLOR

### 5.3.2 Použitie PCA analýzy k redukcii deskriptoru FPFHCOLOR

Pri výpočtoch vhodnej redukovanej dimenzii $L$ (viď kapitola 3.8.7) som použil prahovú hodnotu 95% (tzn. toľko percent pôvodných informácií bude zachovaných).

Na základe tohto prahu algoritmus vypočítal hodnotu **22**. To znamená, že redukovaním pôvodných 66 dimenzií na 22 stratíme iba 5% pôvodných informácií.

Tento vypočítaný redukovaný počet dimenzií FPFHCOLOR som použil následne k rozpoznaniu všetkých testovacích dátových sád. Všetky objekty boli úspešne rozpoznané pri použití takto redukovaných deskriptoroch. Počet nájdených korešpondencií sa zvýšil v priemere o 40%, ale klesol počet korešpondencií patriacich do výslednej transformácii v priemere o 50%.

V porovnaní s pôvodnými FPFH deskriptormi si redukovaná, 22-dimenzionálna verzia FPFHCOLOR viedla na farebnejších scénach lepšie, keďže počet správnych korešpondencií bol stále viac ako o polovicu väčší.

K prípadnému použitiu redukovaných FPFHCOLOR deskriptorov nahráva aj čas strávený redukciou dimenzií. V prípade, že sa v kroku, kedy sa vytvára databáza modelov (viď kapitola 4.4) vykoná PCA analýza a získajú sa bázové vektory (analýza sa vykonáva iba raz, pri spustení programu), tak následná redukcia deskriptorov scény, ktorá môže byť v praxi variabilná (bude sa obraz snímať a spracovávať v reálnom čase), bude trvať okolo 3 ms. Takto strávený čas nám potom zjednoduší párovanie deskriptorov. Viac o zrýchlení tohto kroku a dosiahnutých experimentálnych výsledkov je dostupný v kapitole 5.1.4.

## 5.4 Párovanie deskriptorov pomocou BruteForce

Pri realizovaní kroku, v ktorom sa hľadajú korešpondujúce dvojice bodov z modelu a zo scény (viď kapitola 3.5) sa naskytla dôležitá otázka, ktorá sa týkala efektívnosti použitia dátovej štruktúry k-d strom (viď kapitola 3.7). V tomto kroku sa k-d strom používa pre vyhľadanie $k$ najbližších susedov (v deskriptoroch modelu) aktuálne spracovávaného bodu zo scény. Pri vysokom počte dimenzií deskriptorov sa používajú aproximačné metódy, ktoré sú bližšie popísané v kapitole 5.1.

Druhým spôsobom vyhľadania $k$ najbližších susedov je využitie algoritmu hrubej sily (tzv. BruteForce), ktorý pri spracovávaní deskriptoru zo scény prechádza všetky deskriptory z modelu a hľadá najbližších $k$ susedov na základe vypočítanej Euklidovskej vzdialenosti medzi dvoma bodmi. Teoretická časová zložitosť vyhľadávania je $O(log(n))$.

Aj keď v niektorých prípadoch (napr. vyššia dimenzia deskriptorov) k-d strom vykazuje spomalenie, časová zložitosť BruteForce algoritmu sa blíži k $O(n)$, preto bude sériové vykonávanie vždy pomalšie ako k-d strom. To sa však dá vylepšiť paralelizáciou algoritmu hrubej sily.

Hlavnou myšlienkou tohto vylepšenia je teda navrhnúť algoritmus hrubej sily, paralelizovať ho pomocou frameworku OpenCL a porovnať časy behov pri sériovom a paralélnom spracovaní. Pre tieto potreby som vytvoril šablónu triedy, ktorú som nazval `BruteForceCorrespondences`. Základ šablóny je inšpirovaný algoritmom, ktorý je implementovaný v knižnici PCL a slúži na

podobné účely (hľadanie korešpondencií hrubou silou), ale pracuje iba so štruktúrou reprezentujúcou bod so súradnicami X,Y,Z.

Moja šablóna pracuje podobným spôsobom ako `GenericPCA`, takže je možné použiť všetky typy deskriptorov, ktoré majú definované chovanie spôsobom popísaným v kapitole 5.1.3.

### 5.4.1 Výsledky BruteForce

Vyvinutý algoritmus hrubej sily som použil pre porovnanie časov s inými algoritmami, konkrétne:

- k-d strom z knižnice PCL
- k-d strom s aproximačnými metódami z knižnice FLANN

Meranie prebiehalo na dátovej sade zubnej pasty. Scéna obsahovala 1008 deskriptorov a model 315. Výsledky testovania sú prezentované v tabuľke 8. Je z nich patrné, že všetky testované metódy boli rýchlejšie ako mnou implementovaný algoritmus hrubej sily. Toto zistenie by zodpovedalo časovým zložitostiam jednotlivých algoritmov. Neprejavilo sa ani spomínané zhoršenie klasického k-d stromu.

| Názov algoritmu | Čas behu * |
|---|---|
| Hrubá sila (BruteForce) | 87 ms |
| k-d strom (PCL) | 8 ms |
| k-d strom (FLANN) | 7 ms |

* zaznamenaný 5 krát a následne spriemerovaný

Tabuľka 8: Porovnanie algoritmu hrubej sily s k-d stromom implementovaným v knižniciach PCL a FLANN

Z časových dôvodov som nebol schopný paralelizovať algoritmus hrubej sily. V prípade pokračovania na tejto práci by to bol rozumný krok, ktorý by som vykonal, pretože výsledky behu na architektúre CUDA alebo pomocou frameworku OpenCL by boli odlišné od sériového spracovania.

## 5.5 Dosiahnuté časy rozpoznávacieho reťazca

Ako bolo v tejto práci spomenuté, rozpoznanie objektu v RGB-D obraze (na základe modelu hľadaného objektu) pracuje v krokoch, ktoré na seba naväzujú. Tieto kroky tvoria rozpoznávací reťazec, na konci ktorého sa nájde správna transformácia objektu v scéne.

Moje modifikácie a experimenty sa týkali niektorých krokov a algoritmom, ktoré sa v nich používajú. Meranie časov bolo vykonané v rámci modifikovaného kroku, ale nie v rámci celého reťazca krokov. Preto sú v tabuľke 9 prezentované celkové časy, za ktoré bol objekt rozpoznaný. Pri meraní som použil model a scénu valca, kde spracovávaná scéna obsahuje 29706 bodov. Je použitých niekoľko variánt deskriptorov. Následne je prepočítaná výkonnosť v jednotkách fps (obrazová frekvencia), ktorú reťazec dosahuje.

Je možné vidieť že najväčšiu obrazovú frekvenciu poskytuje redukovaný FPFH deskriptor. Môj deskriptor, ktorý rozširuje FPFH o farebnú zložku RGB (FPFHCOLOR) je síce presnejší ako pôvodný deskriptor, ale dosahuje horších časov a zrýchlenie redukciou dimenzií nebolo až tak veľké, ako u pôvodného FPFH.

Všetky uvedené merania sú vykonané sériovým spracovaním algoritmov, preto existuje priestor pre zlepšenie týchto časov prípadnou paralelizáciu na GPU.

| Názov deskriptoru | Použitá redukcia | Čas rozpoznávania objektu [a] | Obrazová frekvencia (fps) [b] |
|---|---|---|---|
| PFH | Nie | 207 ms | 5 |
| | Áno | 146 ms | 7 |
| FPFH | Nie | 157 ms | 7 |
| | Áno | 114 ms | 9 |
| PFHRGB | Nie | 228 ms | 5 |
| | Áno | 204 ms | 5 |
| FPFHCOLOR | Nie | 208 ms | 5 |
| | Áno | 191 ms | 5 |
| SHOT | Nie | 511 ms | 2 |

[a] medián získaných hodnôt pri 5 testoch
[b] zaokrúhlená na celé čísla nahor

Tabuľka 9: Časy rozpoznávania objektu valca pri použití rôznych variánt algoritmov

# 6 Záver

Vďaka tejto práci som sa zoznámil s problematikou rozpoznávania objektov v RGB-D obraze na základe modelov týchto objektov. Výsledkom rozpoznávania bola správna rotačná matica a vektor posuvu modelu v scéne, čo viedlo k získaniu 6 DoF pozície objektu.

V prvom kroku som vytvoril aplikáciu, ktorá slúži na rozpoznanie objektu v scéne. Poskytuje niekoľko parametrov príkazového riadku, vďaka ktorým je možné proces meniť a využívať rôzne State of the Art algoritmy (vďaka použitiu knižnice PCL), slúžiace pre popis geometrie v okolí bodu, nazývané deskriptory. Celý proces vývoja aj s vyhodnotením týchto algoritmov som popísal v kapitole 4. Zistil som, že kvôli nízkemu počtu dimenzií a dosahovaných časov je deskriptor FPFH dobrým kandidátom na použitie v experimentoch.

Následne som navrhol niekoľko vylepšení celého procesu tak, aby bol rýchlejší, resp. presnejší. Prvou modifikáciou bola redukcia dimenzií počítaných deskriptorov (viď kapitola 5.1), ktorá prispela k zjednodušeniu párovania deskriptorov. Tento krok mi taktiež ukázal, že väčšina dimenzií v odhadnutých deskriptoroch nenesie žiadnu dôležitú informáciu a aplikovanou redukciou je možné niekoľkonásobne zrýchliť proces párovania deskriptorov so zachovaním správneho výsledku.

V druhej modifikácii som zredukoval pôvodný počet dimenzií deskriptoru FPFH a zistil som, že algoritmus dokáže správne odhadovať objekt v scéne aj s pomocou menšieho počtu dimenzií (viď kapitola 5.2). Tieto výsledky sú samozrejme získané iba testovaním na mojich dátových sadách, takže toto tvrdenie nemusí platiť všeobecne.

Následne som na základe deskriptoru PFHRGB, ktorý okrem geometrie v okolí bodu zachytáva taktiež farebné zložky RGB, modifikoval deskriptor FPFH tak, aby kódoval farebnú zložku RGB rovnakým spôsobom, ako to robí algoritmus PFHRGB. Dosiahnuté výsledky na farebných scénach sú presnejšie, ako varianta bez farebnej zložky (viď kapitola 5.3.1). PCA analýzou som taktiež zistil, že takto modifikovaný algoritmus je možné redukovať z 66 dimenzií na 22 dimenzií (zníženie o $\frac{2}{3}$ dimenzií) pri zachovaní 95% pôvodných informácií. Preto si myslím, že farebná zložka má svoj význam a nebola pri popisovaní okolia zbytočná. V opačnom prípade by bolo možné zredukovať počet dimenzií deskriptoru FPFHCOLOR na podobnú hodnotu, aká vyšla pre FPFH.

Mojou motiváciou nakoniec bolo vyvinúť a paralelizovať algoritmus hrubej sily, hľadajúci korešpondencie medzi bodmi a následne porovnať výsledky behu s k-d stromom, ktorý sa v praxi bežne používa. Algoritmus som vytvoril a testoval pri sériovom vykonávaní. Bohužial z časových dôvodov som nestihol algoritmus paralelizovať, preto výsledky pri paralélnom spracovaní nie sú dostupné.

Poslednú časť mojich experimentov tvorí zhodnotenie časovej náročnosti celého procesu rozpoznávania objektu pri použití rôznych variánt deskriptorov (bez redukovania dimenzií a s redukovaním). Tomuto zhodnoteniu je venovaná kapitola 5.5. Meraním som zistil, že pri použití

redukovaných deskriptorov zlepšíme čas vykonávania v každom testovanom prípade. Toto zlepšenie nastáva kvôli zefektívneniu kroku nazývaného párovanie deskriptorov.

Na druhej strane je tu ešte veľký priestor pre zlepšenie, keďže najlepšia nameraná varianta dokáže spracovávať mračná bodov s obrazovou frekvenciou okolo 9 fps, čo je pri použití v aplikáciách, ktoré musia pracovať v reálnom čase, stále málo.

K tomuto by mohla pomôcť paralelizácia niektorých algoritmov (napr. na GPU), ktoré v tejto práci používam. Z toho dôvodu by tento krok bol v budúcnosti ďalším, ktorému by som sa venoval. Dôvodom je aj zistenie, že väčšina používaných algoritmov môže byť efektívne paralelizovaná. Okrem toho by som chcel vyskúšať namiesto farby RGB kódovať farebnú zložku vo farebnom modele HSV, keďže je tento model menej citlivý na zmenu osvetlenia.

# Literatúra


[1] Point cloud, *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2017-03-30].
Dostupné z: `http://en.wikipedia.org/wiki/Point_cloud`

[2] RGB Color Model, *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2017-03-30].
Dostupné z: `https://en.wikipedia.org/wiki/RGB_color_model`

[3] The PCD (Point Cloud Data) file format, *Point Cloud Library* [cit. 2017-03-30].
Dostupné z:
`http://pointclouds.org/documentation/tutorials/pcd_file_format.php`

[4] Pearson, K. (1901), *On Lines and Planes of Closest Fit to Systems of Points in Space.* Philosophical Magazine. 2 (11): 559–572.

[5] Delaunay, Boris (1934). *Sur la sphère vide*, Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles. 6: 793–800.

[6] R. Azuma. *Presence: Teleoperators and Virtual Environments* , 6(4):355–385, August 1997

[7] E. Paquet, M. Rioux, A. Murching, T. Naveen, and A. Tabatabai, *Description of shape information for 2-D and 3-D objects*, Signal Process.: Image Commun., vol. 16, no. 1, pp. 103–122, 2000

[8] Rusu, R. B., *Semantic 3D Object Maps for Everyday Manipulation in Human Living Enviroments*, Dizertačná práca, der Technischen Universität München eingereicht und durch die Fakultät für Informatik, 2009, [cit. 2017-04-04].

[9] R. B. Rusu, N. Blodow, and M. Beetz, *Fast point feature histograms (FPFH) for 3D registration*, in Proc. IEEE Int. Conf. Robot. Autom., 2009, pp. 3212–3217.

[10] Rusu, G. Bradski, R. Thibaux, and J. Hsu, *Fast 3D recognition and pose using the viewpoint feature histogram*, in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2010, pp. 2155–2162.

[11] Six degrees of freedom, *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2017-04-05].
Dostupné z: `https://en.wikipedia.org/wiki/Six_degrees_of_freedom`

[12] R.B. Rusu, N. Blodow, Z.C. Marton,m. Beetz, *Aligning Point Cloud View using Persistent Feature Histograms* In 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, 2008, [cit. 2017-02-04].

[13] R. B. Rusu and S. Cousins, *3D is here: Point Cloud Library (PCL)* , Proc. IEEE Int. Conf. Robotics and Automation (ICRA), Shanghai, China, May 9

[14] G. Guennebaud, B. Jacob, et al. *Eigen v3*, 2010
Dostupné na: `http://eigen.tuxfamily.org`

[15] M. Muja and D. G. Lowe, *Fast approximate nearest neighbors with automatic algorithm configuration*, in International Conference on Computer Vision Theory and Application VISSAPP'09) INSTICC Press, 2009, pp. 331–340.

[16] W. Schroeder, K. Martin, and B. Lorensen, *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*, Kitware, December 2006.

[17] Falahati Soroush, *OpenNI Cookbook*, Packt Publishing Ltd, 2013.

[18] Bessel's correction, *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2017-04-12].
Dostupné z: `https://en.wikipedia.org/wiki/Bessel's_correction`

[19] A. Aldoma et al., *Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation,* v IEEE Robotics & Automation Magazine, vol. 19, no. 3, pp. 80-91, Sept. 2012, [cit. 2017-04-22].

[20] Hough Transform, *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2017-04-22].
Dostupné z: `https://en.wikipedia.org/wiki/Hough_transform`

# A Prílohy

Súčasťou tejto práce je aj DVD nosič, ktorý obsahuje mnou vyvinutý software na rozpoznávanie objektov (na základe ich modelov) v RGB-D obraze. Nosič obsahuje aj použité dátové sady, ktoré sa nachádzajú v zložke `data`.

## A.1 Použitie knižnice PCL

Knižnica PCL používa multiplatformový CMake, ktorý obsahuje konfiguračné súbory CMakeFiles.txt. Praktickú ukážku tohto súboru je možné nájsť v mojej aplikácii. Inštaláciu knižnice PCL je možné vykonať viacerými spôsobmi, konkrétne:

- inštaláciou z repozitáru (resp. predkompilovaných binárnych súborov)
- kompiláciou zdrojových kódov

Pri vývoji som skúsil inštaláciu oboma spôsobmi, ale pri druhej variante nastávali chyby, takže som nakoniec úspešne nainštaloval knižnicu PCL vo verzii 1.7.1 z repozitáru distribúcie Ubuntu, použitím nasledujúcich príkazov:

- `sudo apt-get install build-essential cmake libpcl1.7 libpcl-dev pcl-tools`
- `sudo apt-get install libproj-dev`
- ak by stále nebolo možné spustiť projekt, musí byť do súboru CMakeLists.txt pridaný riadok s textom: `list(REMOVE_ITEM PCL_LIBRARIES "vtkproj4")`

## A.2 Použitie aplikácie na rozpoznávanie objektov

Aplikácia, ktorú som implementoval ako časť tejto práce, slúži na rozpoznávanie objektov (ich 3D pozície) na základe modelov v RGB-D obraze. Konkrétne sa jedná o dve mračná bodov, ktoré reprezentujú model objektu a scénu, v ktorej sa bude následne objekt hľadať.

Jedná sa o konzolovú aplikáciu vyvinutú v programovacom jazyku C++. Používa externé knižnice, medzi ktoré patrí mimo iné knižnica Point Cloud Library (viď kapitola 4.1), ktorá slúži k spracovávaniu mračien bodov. Pre správne spustenie aplikácie je nutné vykonať inštaláciu knižnice PCL jedným z postupov, ktoré sú detailnejšie popísané v prílohe A.1. Doporučenou variantou je inštalácia z repozitáru, v ktorom sa aktuálne nachádza verzia 1.7.1.

Keďže sa jedná o konzolovú aplikáciu, tak všetky parametre, vrátane ciest k mračnám bodom reprezentujúcim snímanú scénu a model objektu, sú predávané parametrami príkazového riadku pri samotnom spustení.

Spustením aplikácie bez žiadneho parametru, alebo s parametrom `-h`, sa objaví zoznam všetkých príkazov aj s vysvetleniami. Všetky frázy a časti kódu sú napísané kvôli konzistencii v anglickom jazyku.

Základné použitie programu je zobrazené vo výpise 1.

```
./RGBD_Pose_Estimation model_filename.pcd scene_filename.pcd [Options]
```

Výpis 1: Základné použitie aplikácie na rozpoznávanie objektov

Voliteľné parametre, ktoré sa môžu dosadiť do časti `[Options]`, sú popísané v tabuľke 10. Je nutné dodať, že aktuálne su v aplikácii dostupné dva algoritmy deskriptorov: FPFH a FPFHCOLOR. Pri použití parametru `-rgbfpfh` bude použitá farebná varianta, inak klasický FPFH. Pridaním parametru `-pca` bude použitá redukcia na zvolený deskriptor (pri FPFH na 8 dimenzií, pri FPFHCOLOR na 22 dimenzií).

| Parameter | Popis parametru | Doporučená hodnota |
|---|---|---|
| -h | Zobrazí pomocníka | - |
| -k | Vykreslí použité kľúčové body | - |
| -c | Vykreslí nájdené korešpondencie medzi modelom a scénou | - |
| -rgbfpfh | Budú použité deskriptory FPFHCOLOR | - |
| -pfh | Budú použité deskriptory PFH | - |
| -pfhrgb | Budú použité deskriptory PFHRGB | - |
| -pca | Budú redukované deskriptory definované jedným z troch vyššie uvedených parametrov (ak nie je vybraný žiadny, bude vykonaná redukcia FPFH) | - |
| -shot | Budú použité neredukované deskriptory SHOT | - |
| –algorithm | Výber zhlukovacieho algoritmu. Povolené hodnoty: Hough alebo GC | GC |
| –model_ss | Veľkosť listu pri podvzorkovaní modelu | 0.01 |
| –scene_ss | Veľkosť listu pri podvzorkovaní scény | 0.03 |
| –rf_rad | Veľkosť referenčných rámov pri použití Houghovho hlasovania | 0.015 |
| –descr_rad | Okolie (v metroch) s ktorým sa budú počítať deskriptory | 0.009 |
| –cg_size | Veľkosť zhluku | 0.01 |
| –cg_thresh | Minimálny počet bodov v zhluku | 5 |
| –kd_tree_thresh | Maximálna vzdialenosť medzi bodmi pri párovaní deskriptorov | 50 |
| –num_k | Počet hľadaných susedov pri párovaní deskriptorov | 1 |
| –normal_k | Počet susedov zahrnutých do výpočtu normál | 16 |
| –retain_var | Číslo v intervale $[0.0, 1.0]$ reprezentujúce percento zachovaných informácií pri výpočte redukovanej dimenzie zobrazenej v grafickom okne | 0.95 |

Tabuľka 10: Voliteľné parametre aplikácie k rozpoznávaniu objektov

## A.3 Ukážky práce s triedou GenericPCA

Základne použitie šablóny triedy, ktorá je určená pre vykonanie PCA analýzy na štruktúrach definovaných v knižnici PCL, je zobrazené vo výpise 2. Pred samotným použitím šablóny triedy je nutné zaregistrovať nový dátový typ v rámci knižnice PCL. To je vykonané pomocou makra `POINT_CLOUD_REGISTER_POINT_STRUCT`. Následne je ešte pomocou kľúčového slova `typedef` vytvorený nový dátový typ, ktorý reprezentuje výstupný typ bodu.

Algoritmus pracuje s dvomi mračnami bodov, ktoré reprezentujú vždy model a scénu. Po

zavolaní funkcie `setInputModel` je mračno transformované do matice, z ktorej je následne vypočítaný vektor stredných hodnôt, kovariančná matica, vlastné čísla, vektory a následne vektor súhrnných energií. Je možné teda použiť funkciu `getSuggestedDimensions`, ktorá na základe poskytnutého percenta vypočíta novú redukovanú dimenziu pre vstupné dáta.

Ďalším krokom je načítanie mračna bodov reprezentujúceho scénu (funkcia `setInputScene`), nasledované volaním funkcie `setReducedDimensions`, ktorá vytvorí novú bázu vektorov. Báza bude použitá po zavolaní funkcie `compute` pre redukciu dát. Na získanie výsledných redukovaných mračien bodov je nutné zavolať funkcie `getModelPoints`, resp. `getScenePoints`.

```
POINT_CLOUD_REGISTER_POINT_STRUCT (pcl::Histogram<8>,
        (float, histogram, histogram)
)

typedef pcl::Histogram<8> PCAHistogram;

pcl::PointCloud<PCAHistogram>::Ptr reduced_model_descriptors(new pcl::
    PointCloud<PCAHistogram>());
pcl::PointCloud<PCAHistogram>::Ptr reduced_scene_descriptors(new pcl::
    PointCloud<PCAHistogram>());

GenericPCA<pcl::FPFHSignature33, PCAHistogram> pca_analysis(33);

pca_analysis.setInputModel(*model_descriptors);
pca_analysis.setInputScene(*scene_descriptors);
pca_analysis.setReducedDimensions(8);

pca_analysis.compute();

*reduced_model_descriptors = pca_analysis.getModelPoints();
*reduced_scene_descriptors = pca_analysis.getScenePoints();
```

Výpis 2: Základné použitie aplikácie šablóny triedy GenericPCA